About uICE

Overview

uICE enables complete control of your Windows system with handheld remote controls, standard and multimedia keyboards, wireless or wired. Use uICE to remote-control multimedia applications like Windows Media Player, Winamp or other MP3 players, your favorite DVD/DivX video player or TV viewer, or presentation software like Microsoft Powerpoint.

uICE is a versatile all-in-one interface between an input device and any kind of software application. Optimized for controlling a Windows PC from the distance, it provides On-Screen Displays (OSDs) and an On-Screen Menu (OSM) known from TV sets and DVD/VCR devices. To ensure readability on TV sets and smaller computer displays even from a greater distance, OSDs and the OSM renounce displaying a lot of graphical elements in favor of customizable large fonts.

Its versatility in application and system control comes from its large collection of actions both for common Operating System procedures and application input simulation, especially its advanced and powerful PostMessage action and the convenient On-Screen Menu.

In addition, uICE provides true multi-application remote control, which allows to define independent sets of remote control functions for each application. The Play button can play videos, music, slide shows or whatever you like, depending on the currently selected application.

Application selection is, of course, accessible via the On-Screen Menu, which does also provide options to explore and open videos, music, pictures and other files directly from the harddisk or a home network/Intranet location, and timers to turn your computer off or wake it up from standby mode at a specific time.

Using a plugin driver technology for hardware access, uICE can be easily extended to support any kind of remote control receiver or other input hardware. With its built-in network server, it is even possible to develop client applications with native remote control support without accessing the hardware itself.

Turn your PC into an all-in-one Multimedia Home-Theater system.

System requirements

- Windows 98/ME, Windows 2000 or Windows XP
- Internet Explorer 5.0 or later

For remote control functionality (Keyboard-only mode does not require special hardware):

- Supported receiver hardware
- Remote control

Windows NT based systems require user accounts with rights to start and stop services (usually at least Power User accounts).

How to register

Online registration

Online registration is available through the ShareIt! registration service. We accept all major credit cards like Mastercard/Eurocard, Visa, American Express and Diner's Club online via a VeriSign secured SSL web interface. Credit card orders are usually processed within a maximum of 48 hours.

Offline registration

If you don't like to use your credit card number online, you have the option to use bank transfer/wire transfer, Checks or Money Orders through postal mail. Just fill in the secure form and select your preferred payment method, or follow the link to place a fax order at the bottom of the page.

Why registering?

The unregistered shareware version refuses to run after 30 days. There's no need to download and/or install uICE again after the end of the trial period, and you won't need to reconfigure it after registering. You will receive detailed registration instructions by email which will unlock your existing installation.

Legal Information

END-USER LICENSE AGREEMENT FOR UICE

IMPORTANT - READ CAREFULLY: This End-user License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and the copyright holder for one (1) license to use uICE. uICE includes computer software and associated media and printed materials, and may include online or electronic documentation ("the product" or "the software"). By installing, copying, or otherwise using the software, you agree to be bound by the terms of this EULA.

- All copyrights to the software are exclusively owned by the author, Christian Mueller ("copyright holder").
- This software is provided "AS IS". Although it is thoroughly tested, it may contain hidden bugs. The author disclaims all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The author assumes no liability for damages, data loss, loss of profits or any other kind of loss, direct or consequential, which may result from the use/misuse, the unability to use or the installation of this software. The functionality described in the documentation is not garantueed and should be evaluated by each user in his own computer environment before registering.
- Anyone may use the software during a trial period of 30 days. Following this trial period of 30 days or less, if you wish to continue to use the software, you MUST register. Otherwise, all existing copies of the software must be immediately removed from the system.
- Once registered, one (1) user is granted a non-exclusive license to use the software on up to five (5) computers, OR several users are granted a non-exclusive license to use the software on one (1) computer (both refered as "single-user license"), for any legal purpose. The registered software may not be rented or leased.

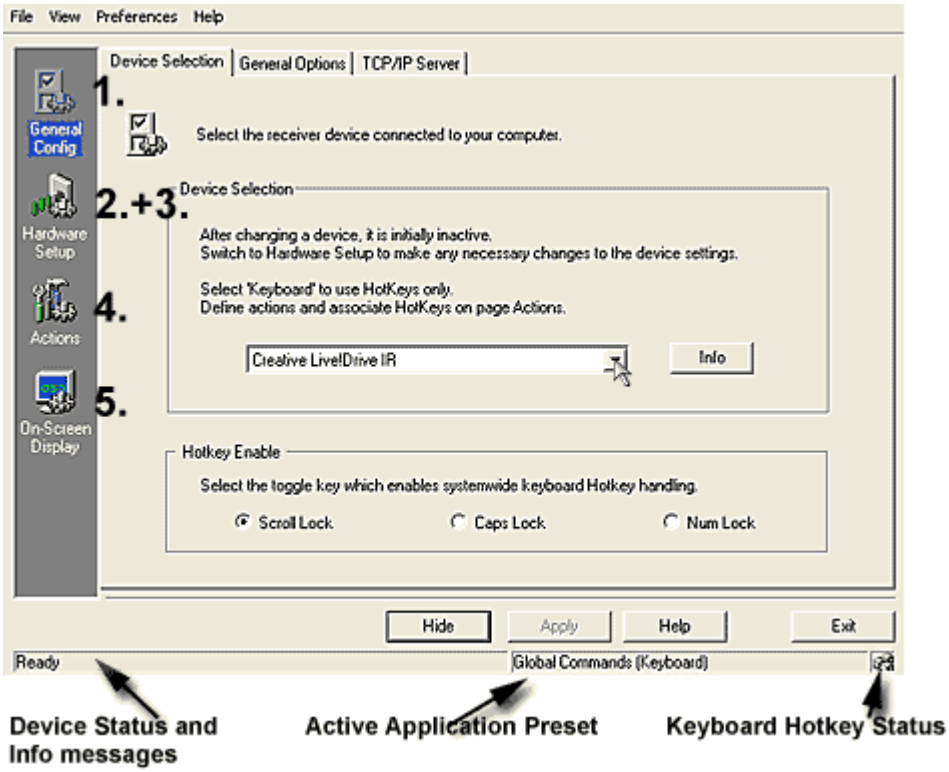
- You may also store or install a copy of the software on a storage device, such as a network server, used only to install or run the software on your other computers over an internal network. However, you must acquire and dedicate a license for each separate computer on which the software is installed or run from the storage device.
- The unregistered shareware version of the software may be freely distributed, provided the distribution package is not modified. No person or company may charge a fee for the distribution of the software without written permission from the copyright holder.
- You may not use, copy, emulate, clone, rent, lease, sell, modify, decompile, disassemble, otherwise reverse engineer, or transfer the software, or any subset of the software, except as provided for in this agreement. Any such unauthorized use shall result in immediate and automatic termination of this license and may result in criminal and/or civil prosecution.
- Installing the software signifies acceptance of these terms and conditions of the license.
- If you do not agree with the terms of this license you must remove all copies of the software from your storage devices and cease to use the product.

Quickstart

uICE features a Setup Guide which explains the basic concepts of the user interface and setup steps. To start the guide, select Help|Setup Guide from the uICE main menu.

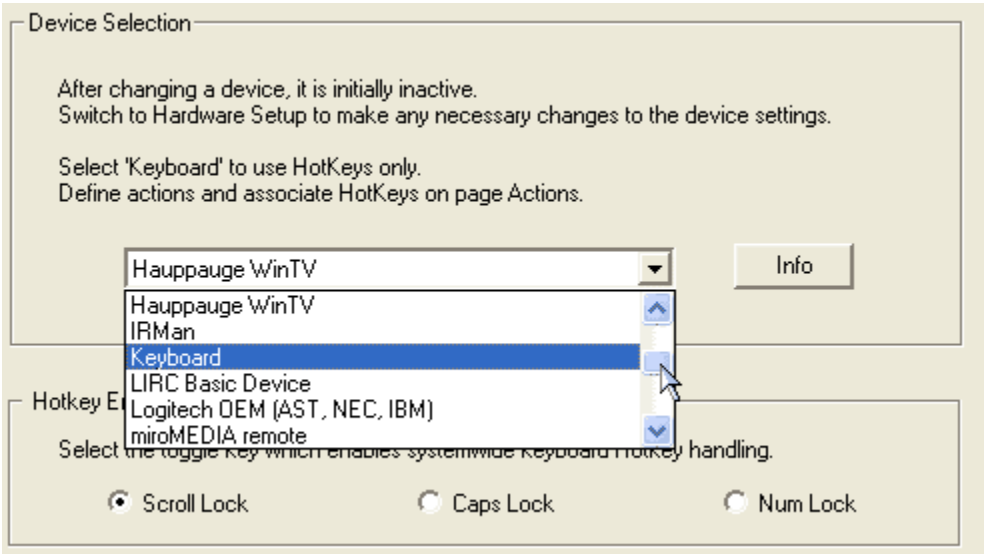
For more detailed information, follow the steps below.

1. Select your hardware
2. Configure the hardware
3. Learn the buttons of your remote control
4. Associate actions with the buttons
5. Setup the On-Screen Menu



Device Selection

This should be your first step in setting up uICE. Select your remote control receiver device from the list. The *Info* button pops up information as well as configuration tips about the selected driver.



If you'd like to use Hotkeys without a remote control, select Keyboard.

If you'd like to use both a keyboard and a remote control, select your receiver device from the list. The Hotkey functionality is always available.

The *Hotkey Enable* toggle key enables or disables keyboard Hotkey handling. If you select the Scroll Lock key for example, Hotkeys are active if Scroll Lock is on. If your keyboard has an indicator light, it will reflect the current status. Additionally, uICE shows the status in the status bar:

- Hotkeys enabled:
- Hotkeys disabled:

Using the toggle key, you can switch your keyboard into remote control mode and back into standard keyboard mode for text processing for example, just as required. This functionality lets you use all keys for remote control actions without limiting normal typing.

After selecting the device, switch to Hardware Setup to learn the buttons of your remote control.

General Options

Customize the general behaviour of uICE. All options are per-user settings.

Start with Windows

Enable this checkbox to let uICE automatically start with Windows.

Hide on startup

Enable this checkbox to automatically hide the main window when uICE starts up.

Disable exit confirmation

By default, uICE asks if you really want to exit when you click the Exit button. Exiting uICE means that all remote control functions as well as any active sleep- or wake-up timers get disabled.

Enable this checkbox to disable the question.

Display shutdown window

uICE needs to perform several deinitialization steps to shut down. By default, a progress window is displayed when shutting down. If this checkbox is enabled, the window won't be displayed, but uICE still needs a few seconds to terminate.

Disable multithreaded execution

uICE makes extensible use of multithreading to execute commands. On slower machines, enabling this option may improve performance, but makes the user interface unresponsive while executing commands, especially if command sequences with delays are in the queue.

Enable this option to disable execution multithreading.

Boost priority on execution

Enable this checkbox to boost priority on execution.

On slower machines, this will improve the response time when pressing a button on the remote, but may also affect other applications that require much CPU resources, like software DVD playback.

Check for updates automically

uICE can check the Internet for updates automatically. If enabled, uICE looks for updates weekly if you are online. If you're not online, the update check is postponed, uICE won't dial up itself.

Leave window focus after keystroke

The basic keystroke actions (Keystroke to application, Keystroke to window) require a window (or application) to be put into foreground before it can accept a keystroke. uICE uses a mechanism to restore the window order on the desktop to its previous state a few seconds after the keystroke, which has certain performance issues on slower machines.
Enable this checkbox if you don't like this feature, or use PostMessage instead, which is able to simulate keystroke messages without changing the window order on the desktop.

Enable forced system shutdown

If checked, uICE instructs Windows to not ask applications to save open documents on system shutdown. Check this box if you want to ensure that the system is turned off without additional user input when you press the Power button on your remote control.

Warning: *Forced* means that applications with unsaved documents will **not display any warnings or questions reminding you to save the documents**, i.e. unsaved data will be lost.

Auto-activate foreground application

If checked, uICE automatically activates and deactivates application presets according to the current foreground window

"Repeated command timout" and "Typematic delay" are now properties of the remote control at Hardware Setup .

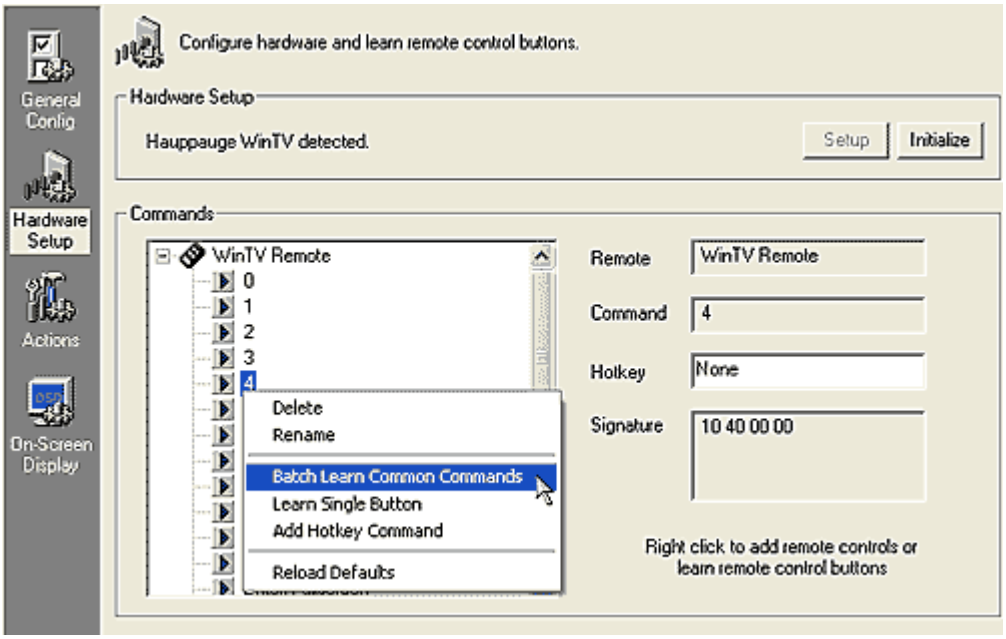
TCP/IP Server

Configure the built-in TCP/IP server. Usually you won't need to adjust these settings, except for three cases:

- You want to use a remote control client (e.g. the Winamp plugin) on another computer located somewhere in your home network or the Internet. In this case, enable the Make server available to other machines checkbox.
- Another application is using the server's default port, preventing the server from starting up (you'd see a message in the status bar). In this case, change the TCP/IP server port value in uICE *and* the client application.
- The Enable log option enables extensive logging of all server transactions and possible errors to the file svrex.log in the uICE installation folder. If you experience connections problems between a client and the server, examining this file may help to track down network problems.

Hardware Setup

After you've selected your receiver device at General Config, you can configure the hardware and learn the buttons of your remote control or define keyboard Hotkeys at Hardware Setup. The figure below shows the setup screen for TV capture boards, note that the upper part may look different for your receiver device.



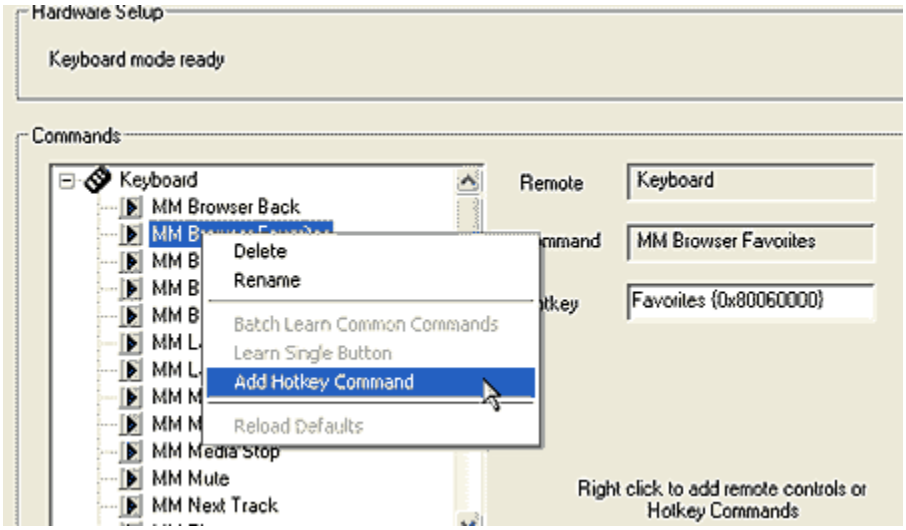
Configure your hardware and click *Initialize*. For help on driver settings, click the *Info* button at General Config|Device Selection or look under the Devices book in this help file.

You can adjust several remote control settings by selecting a remote control's label:

- Typematic
- AntiRepeat

Once the hardware is running, you can start learning buttons or setting up Hotkeys. If you are unsure whether you have your device properly configured, just try to learn to see whether uICE receives input.

Setting up Hotkeys



Hotkeys are defined at Hardware Setup like remote controls. To add a new Hotkey, open the context menu and select *Add Hotkey Command*. Give the command a unique name and type the key into the *Hotkey* field.

Note: Hotkeys are only active if the *Hotkey Enable* toggle key (Scroll Lock by default) is on! uICE reflects the status in the status bar:

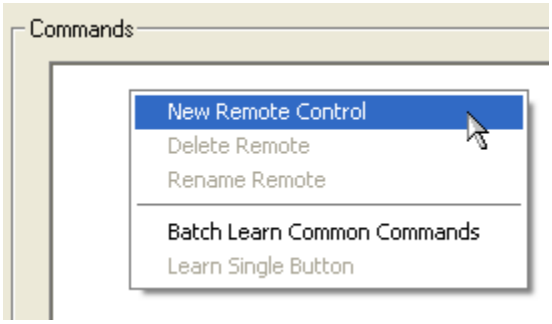
- Hotkeys enabled:
- Hotkeys disabled:

Learning buttons

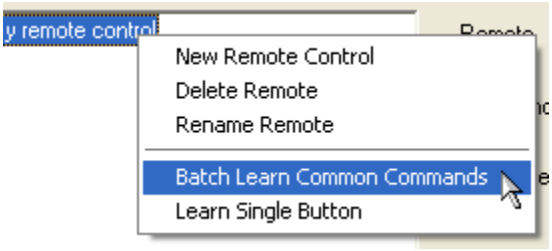
Before you can start using your remote control, you need to learn in the buttons so they can be translated into commands. Note that some devices have their buttons already prelearned, in this case you can skip this topic and proceed to the Actions chapter.

The Commands section provide a context menu to add and edit buttons or remote control folders. Right-click the treeview to open the menu.

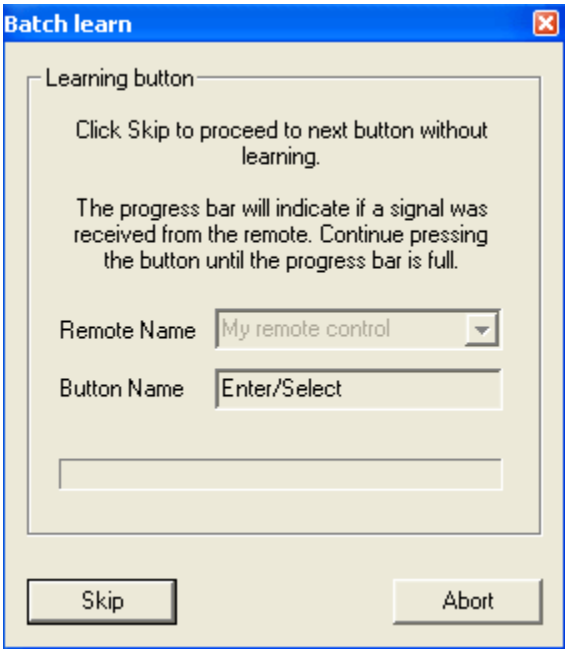
1. Add a remote control folder. From the context menu, select *New Remote Control* .



2. Right-click the remote control label and select *Batch Learn Common Commands* . This will open the Batch Learn dialog. To test whether the hardware is configured properly, you may want to try to learn a single command first by selecting *Learn Single Button* from the context menu.



3. The Batch Learn mode in uICE lets you automatically learn common buttons like Play, Pause and the number buttons. Don't worry if the command names do not match the names of the remote control buttons exactly, you can rename them later. Just hit the button you'd like to have performing the desired action. See Common buttons for a description of the commands. If you don't want to learn a certain button, click *Skip* to proceed to the next one.


Batch learn
Learning button
Click Skip to proceed to next button without learning.
The progress bar will indicate if a signal was received from the remote. Continue pressing the button until the progress bar is full.
Remote Name
My remote control
Button Name
Enter/Select
Skip
Abort

4. After completing the Batch Learn, you can rename or add additional buttons. Use the context menu to add buttons (*Learn Single Button*) and the usual keyboard shortcuts (F2, DEL) to edit them.

If you rename buttons or remote controls at Hardware Setup and have associated actions or On-Screen Menu commands, uICE automatically applies the new name to the affected commands.

Deleting learned buttons or remote controls does not modify any actions.

Common buttons

uICE is partially self-configuring when using the Batch Learn mode shown above. The following table lists the Batch Learn mode buttons and their default functionality. Note that On-Screen Menu commands get only automatically configured if they are not yet occupied.

Menu/Back	Enters On-Screen Menu mode. If the On-Screen Menu is active, goes one level up in the menu hierarchy or turns it off if the main menu is displayed.
Enter/Select	Selects the currently highlighted item from the On-Screen Menu.
Up/Channel+	Moves the On-Screen Menu selection marker up.
Down/Channel -	Moves the On-Screen Menu selection marker down.
Next/Volume +	Switches the currently selected On-Screen Menu item, for example the way the sleep timer suspends your computer. Additionally, uICE automatically creates an action for this button that increases sound volume.
Back/Volume -	Switches the currently selected On-Screen Menu item, for example the way the sleep timer suspends your computer. Additionally, uICE automatically creates an action for this button that decreases sound volume.
Mute	uICE automatically creates an action for this button that mutes the soundcard.
1..0	Used to enter digits in the On-Screen Menu's Sleep and Wake up timers.
Power	uICE automatically creates an action for this button that shuts down the computer.
Play/Pause, Stop, Forward and Rewind	These buttons are used in many application presets.

Now that you've teached uICE to understand the input from your remote control, you can start to configure it to do what you want it to: Actions.

Keyboard

This driver is a generic driver for standard and multimedia keyboards*, wireless or wired. Select this driver if you don't have a supported remote control.

Keyboard Hotkeys are always available, even if you have another remote control device selected.

Important: The *Hotkey Enable* toggle key, configurable at General Config|Device Selection, enables or disables keyboard Hotkey handling. If you select the Scroll Lock key (the default) for example, Hotkeys are active if Scroll Lock is on. If your keyboard has an indicator light, it will reflect the current status. Additionally, uICE shows the status in the status bar:

- Hotkeys enabled:
- Hotkeys disabled:

Using the toggle key, you can switch your keyboard into remote control mode and back into standard keyboard mode for text processing for example, just as required. This functionality lets you use all keys for remote control actions without limiting normal typing.

A tip on **wireless infrared keyboards and universal learning remotes**: you can use many learning remote controls with the keyboard's infrared receiver! Put your remote control into learning mode (consult the remote's manual on how to do this), point it to the keyboard's infrared transmitter (usually at the back of the keyboard), and hit a key. The remote control will learn the signal of the keyboard, and send it out again if your press the remote's button.

***Note**: the default configuration contains multimedia keys for keyboards designed for Windows ME or later. On Windows 98 systems, and when using keyboards that do not follow the Windows standards, it might be required to reassign the multimedia keys to the appropriate hotkey commands.

ATI Remote Wonder

This driver supports the ATI Remote Wonder shipped as add-on for Radeon 7500/8500 and stand-alone.

Configuration

Click the *Initialize* button to detect the hardware.

Important: uICE takes complete control over the remote control, therefore the ATI remote control application (ATIX10.EXE) should not be running at the same time as it would interfere. If a running ATIX10.exe instance gets detected by the driver, it will be closed on device initialization and restarted after exiting uICE.

The Windows driver for the Remote Wonder, available at www.ati.com, has to be properly installed. The "Load on startup" option in ATI's software should be disabled.

Default Actions

The following table shows the default button functions. You can change this functionality at *Actions.*

	Global (no application active)	**Windows Media Player**	**Winamp 3**
	System Shutdown	Close	Close
	Mouse cursor movement		
	Left/Right mouse button		
TV	n/a		
DVD	Launch default media player		

WEB	Launch default browser / Go to homepage		
	n/a		
	Toggle left mouse button for Drag&Drop		
	Volume up/down		
	Previous/Next track (requires support in foreground application)	Next/Previous Track (if applicable)	Next/Previous track (if applicable)
MUTE	Mute		
	Numeric keys		
	Show/Hide On-Screen Menu		
	ENTER key		
	Favorites (Internet Explorer)	Preview (WMP 6) View Mode (WMP 7/8)	n/a
	Minimize current window	Toggle fullscreen mode	Minimize current window
	Up/Down/Left/Right arrow key **Menu navigation** in On-Screen Menu mode		
	ENTER key **Menu Select** in On-Screen Menu mode		
	Previous/Next track (requires support in foreground application)	Rewind/Forward (if applicable)	Rewind/Forward
	Play/Pause (requires support in foreground application)	Play/Pause	Play
	Stop (requires support in foreground application)	Stop	Stop
	Play/Pause (requires support in foreground application)	Play/Pause	Pause
	n/a		
	Close current window		
	n/a		

	Cancel (ESC key)	Cancel (ESC key)	Toggle Crossfade
	n/a		
	n/a	Toggle Repeat (WMP 7/8 only)	Toggle Repeat
	n/a	Toggle Shuffle (WMP 7/8 only)	Toggle Shuffle

Logitech OEM (AST, NEC, IBM)

Configuration

Select the COM port where the device is attached. Click the *Advanced* button to enable or disable the mousepad.

Irman

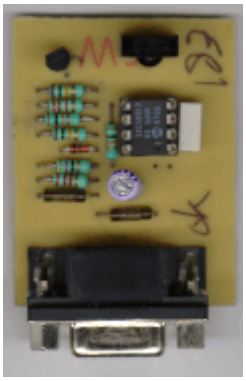
Configuration

Select the COM port where Irman is attached.

More Information

Visit the Irman website

UIR (Universal Infrared Receiver)



The UIR is a home-brew receiver device. With a bit of soldering experience, you can easily build it yourself. Visit the UIR website for circuit diagrams and PCB layouts.

Configuration

Select the COM port where UIR is attached.

More Information

Visit the UIR website

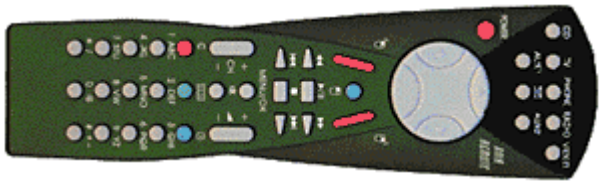
Packard Bell Fast Media

Configuration

Select the COM port where the device is attached.

Note that the Fast Media driver must not be installed.

Anir Multimedia Magic



Configuration

Select the COM port where the device is attached.

Note that the original Animax driver must not be installed.

More Information

Visit the Animax website

Creative Live!Drive IR

This driver supports the Live!Drive IR and AudigyDrive that comes with Soundblaster Live! Platinum and Audigy Platinum soundcards.

Configuration

Click the *Initialize* button to startup the Live!Drive.

Depending on your system's hardware configuration, you may need to switch the driver to another *Input Port*. If uICE doesn't receive input from the remote control, try another port.

More Information

Visit Creative's website

Soundblaster Extigy

This driver supports the Soundblaster Extigy remote control (RM-1000).

Configuration

Click the *Initialize* button to detect the device.

Important: The Windows drivers for the Soundblaster Extigy, available at www.soundblaster.com, have to be properly installed. The RemoteCenter software is *not* required and should be disabled while uICE is running.

Default Actions

The following table shows the default button functions. You can change this functionality at *Actions.*

Note: Power On/Power Off and CMSS are hardware functions and cannot be disabled.

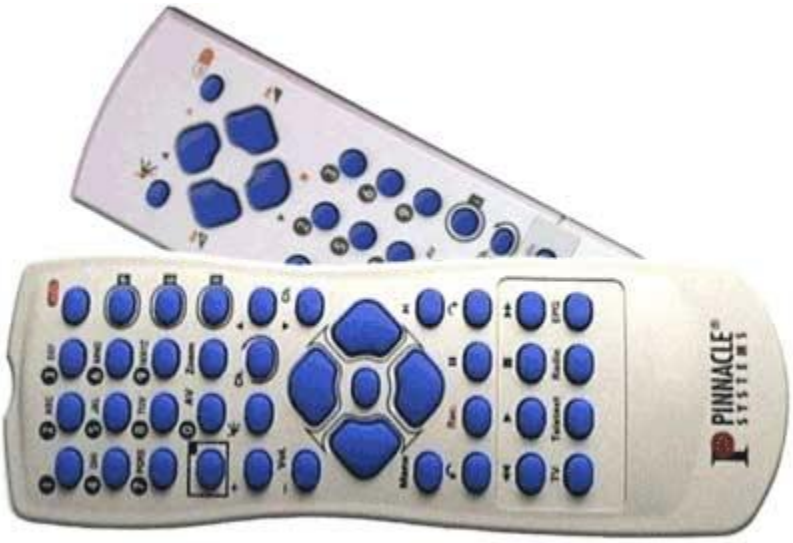
	Global (no application active)	**Windows Media Player**	**Winamp 3**
Power (On/Off)	Extigy On/Off		
1..0	Numeric keys		
CMSS	CMSS On/Off		
Mute	Mute		
REC	System shutdown		
Volume +/-	Volume up/down		
Stop	Stop (requires support in forground application)	Stop	Stop
Play/Pause	Play/Pause (requires support in forground	Play/Pause	Play/Pause

	application)		
Slow	n/a	Toggle Repeat (WMP 7/8 only)	Toggle Repeat
Previous Track Next Track	Previous/Next Track (requires support in forground application)	Previous/Next Track (if applicable)	Previous/Next Track (if applicable)
Step	n/a	WMP 6: Preview WMP 7/8: Toggle Shuffle	Toggle Shuffle
EAX	Audio Mode (Treble, Bass, Panning, Fader)		
Options	Show/Hide OSM File Browser		
Display	Show/Hide OSM Application Selector		
Return	Cancel (ESC key)	Toggle Skin/Full Mode (WMP 7/8 only)	Toggle Crossfade
Start/Menu	**Show/Hide On-Screen Menu**		
Cancel/Close	Close current window	Close	Close
Up Down	Up/Down arrow key	Previous/Next Track (if applicable)	Previous/Next Track (if applicable)
	Menu navigation in On-Screen Menu mode		
Left/Rewind Right/Forward	Left/Right arrow key	Rewind/Forward (if applicable)	Rewind/Forward
	Menu navigation in On-Screen Menu mode		
Select/OK	OK (ENTER key)	Toggle Fullscreen	Pause

More Information

Visit the Soundblaster website

Pinnacle PCTV



This driver supports remote controls from Pinnacle Systems' PCTV product line (PCTV/PCTV Pro/PCTV Sat). Many third-party remote controls should work as well with the PCTV receiver.

Configuration

Select the COM port where the device is attached.

More Information

The default configuration supports the current 41-button remote control. For older PCTV models, you may need to relearn the buttons.

Visit Pinnacle Systems' website

WinFast TV2000 XP Deluxe

This driver supports remote controls from Leadtek WinFast TV2000 XP PCI TV cards.

Configuration

Click the *Initialize* button to detect the hardware.

Important: only the CH+/- and VOL+/- buttons support repeated commands.

Default Actions

The following table shows the default button functions. You can change this functionality at *Actions.*

	Global (no application active)	**Windows Media Player**	**Winamp 3**
Power	System Shutdown	Close	Close
MTS	n/a		
TV/FM	n/a		
Video	Launch default media player		
Display	**Show/Hide On-Screen Menu**		
CH+/-	Up/Down key	Forward/Rewind (if applicable)	Forward/Rewind
	Menu Navigation in On-Screen Menu mode		
VOL+/-	Volume up/down **Menu Navigation** in On-Screen Menu mode		
FullScreen	Show OSM Application Selector	Toggle FullScreen	Show OSM Application Selector
Teletext	n/a		
Sleep	Standby Mode		
BossKey	Close current window		
Mute	Mute		
Red	n/a	Toggle Repeat (WMP 7/8 only)	Toggle Repeat

Green	n/a	Toggle Shuffle (WMP 7/8 only)	Toggle Shuffle
Yellow	n/a	n/a	Toggle Crossfade
Blue	n/a		
0..9	Numeric keys		
FineTune +/-	n/a		
PIP	n/a	WMP 7/8: Skin/Full Mode WMP 6: Preview	n/a
Recall	Cancel (ESC key)		
Enter	OK (ENTER key)		
. (DOT)	n/a		
Back/Next	Previous/Next Track (requires support in forground application)	Previous/Next Track (if applicable)	Previous/Next Track (if applicable)
Play/Pause	Play/Pause (requires support in forground application)	Play/Pause	Play/Pause
TimeShift	n/a		
Stop	Stop (requires support in forground application)	Stop	Stop
REC	n/a		
SnapShot	n/a		

Magic TView (CPH051, CPH061)

This driver supports remote controls from Askey/Anubis/Typhoon Magic TView (CPH051/CPH061) PCI TV cards.

Configuration

Click the *Initialize* button to detect the hardware.

FlyVideo 3000

This driver supports the remote control from Lifeview FlyVideo 3000 (Philips chipset based) PCI TV cards.

Configuration

Click the *Initialize* button to detect the hardware.

Default Actions

The following table shows the default button functions. You can change this functionality at *Actions.*

	Global (no application active)	**Windows Media Player**	**Winamp 3**
Air/CATV	n/a		
Video	Launch default media player		
Audio	n/a		
Power	System Shutdown	Close	Close
Source	Close foreground window	WMP 6: Preview WMP 7/8: Toggle Skin/Full Mode	n/a
Fullscreen	n/a	Toggle fullscreen mode	n/a
Stereo	Audio Mode		
MUTE	Mute		
1	Numeric key	Toggle Repeat (WMP 7/8 only)	Toggle Repeat

2	Numeric key	Previous Track (if applicable)	Previous Track (if applicable)
3	Numeric key	Toggle Shuffle (WMP 7/8 only)	Toggle Shuffle
4	Numeric key	Rewind (if applicable)	Rewind
5	Numeric key	Stop	Stop
6	Numeric key	Forward (if applicable)	Forward
7	Numeric key	n/a	Toggle Crossfade
8	Numeric key	Next Track (if applicable)	Next Track (if applicable)
9	Numeric key	n/a	n/a
0	Numeric key	Stop	Stop
Recall	**Show/Hide On-Screen Menu**		
Enter	**Menu Select** in On-Screen Menu mode		
	Play/Pause (requires support in foreground application)	Play/Pause	Play/Pause
Vol +/-	Volume up/down **Menu Navigation** in On-Screen Menu mode		
Channel +/-	**Menu Navigation** in On-Screen Menu mode		
	Next/Previous track (requires support in foreground application)	Next/Previous Track (if applicable)	Next/Previous Track (if applicable)

TV Capture Board

This is a generic driver which supports many BT8xx based TV capture boards with built-in infrared receiver hardware.

TV cards with an external, serial port based receiver device are not supported by this driver. Use the Generic Serial IR Receiver device instead.

Configuration

Click the *Setup* button to start the configuration wizard and follow the instructions. The wizard configures the driver to understand the signals from your TV card.

For most models it is necessary that the manufacturer's driver and TV viewer application are properly installed.

After you have finished the wizard successfully, you can start learning buttons. The wizard itself does not learn the buttons, it only learns how to interprete the signals.

Hauppauge WinTV

Configuration

Click the *Initialize* button to detect the hardware.

More Information

Visit Hauppauge's website

WinLIRC

The WinLIRC driver does not directly access hardware, but works as a TCP/IP client connecting to the WinLIRC server. For information on how to get WinLIRC and supported hardware, visit the WinLIRC and the LIRC homepage.

WinLIRC Step-By-Step

- Start WinLIRC and learn a remote control as usual.
- Start uICE, select WinLIRC from the dropdown box at *General Config/Device Selection*
- Switch to "Hardware Setup", click the *Setup* button and verify settings for Server IP/URL and Server Port (should be localhost on port 8765). Click *OK* to connect to WinLIRC (make sure WinLIRC is running).
- WinLIRC's Remote and command names are displayed in the white area at the lower half of the Hardware Setup screen. The area gets populated automatically if uICE receives the names from WinLIRC, i.e. if you press the buttons configured in WinLIRC. WinLIRC v0.6.4 does not yet support sending a full list of command and remote names, maybe we'll get this functionality in a later version.
- You can now setup actions as usual.

Configuration

No special configuration is required if WinLIRC runs on the same machine as uICE. However, you may customize port and ip settings if you want to connect to WinLIRC over a network (it may be required that you have to remove address binding to localhost in the WinLIRC sources and recompile it in order to be able to connect to WinLIRC on another machine).

You may even connect to a LIRC daemon on a Linux machine. For this, you need the netpipes package installed on the Linux machine, available from ftp://ftp.purplefrog.com/pub/netpipes/ (probably file netpipes-x.x-export.tar.gz). Install that in the usual way with the help of the included INSTALL and README files and then the command: /path/to/faucet 8765 --out --daemon /path/to/irw - *Thanks to Chris Tallon for this hint.*

Further, you have the option to automatically launch WinLIRC when uICE starts (if WinLIRC is installed on the same computer). Enable the checkbox AutoStart WinLIRC Server and locate the WinLIRC executable file name.

If Reconnection attempt every xx sec is enabled, the driver tries to automatically reconnect to WinLIRC or LIRC in intervals of xx seconds (if not already connected). This is useful if WinLIRC doesn't run on the same machine and you cannot assume that WinLIRC is ready before uICE starts.

More Information

- WinLIRC website
- The LIRC Project

Generic DCD Line Device

This is a generic driver for serial port remote control receivers sending bit streams through the RLSD (**R**eceive **L**ine **S**ignal **D**etect) / DCD (**D**ata **C**arrier **D**etect) line.

One example device using this technology is the basic home-made LIRC / WinLIRC receiver. You can find schematics for this hardware at the LIRC Project website.

All decoding of the bit stream is done in software, therefore accurate timer hardware is required and a fast computer is recommended. High performance timers are present in all GenuineIntel machines with Pentium or higher processors, but there are known problems with other systems, like AMD processors on mainboards with non-Intel chipsets.

Please be aware that not all remote controls will work properly with this driver. Remote controls using standard signal formats (Sony, Philips, Sharp and most other TV/Video/HiFi RCs) were successfully tested, but there are a few remote controls using very uncommon formats or very short signal durations, which your timing hardware can't handle properly.

Configuration

Select the COM port where the device is attached.

More Information

The LIRC Project

LIRC Basic Device

The LIRC Basic Device is a home-brew receiver device. With a bit of soldering experience, you can easily build it yourself. Visit the LIRC Project website for circuit diagrams.

Configuration

Select the COM port where the device is attached.

More Information

The LIRC Project

Generic Serial IR Receiver

The Generic Serial IR Receiver is a generic serial port driver with adjustable settings for devices which conform to the RS-232 protocol standard.

Configuration

Select the COM port where the device is attached. The Setup button opens the driver configuration dialog with the following three property pages:

Port Settings

Adjust the COM port settings.

Device Settings

Init String	Initialization character sequence sent to the receiver
Response	Response character sequence to be received from the receiver after transmitting the *Init String* indicating proper initialization.
IR code length	Length in bytes of a receiver signal for one remote control button depression. If your receiver uses signals of variable lengths (like URCR), use higher values (e.g. 255). Increase the value until signals end with "00".
Tolerance	Adjust this value if your receiver does not always send the same code for one IR signal. This value is a statistical error value without a special meaning. In general, the higher this value, the more variation may appear in the signal, and the more unsafe is the recognition and the risk that a command will be chosen that do not apply to the remote button. Try values between 0 and 16 for safe recognition.
Learn samples	Specify the number of samples taken when learning a button. You may increase this value to have safer code recognition.

Advanced Settings

Hardware timing values in milliseconds. The default settings are fail-safe, but require some time for initialization. Decrease the values to have faster device initialization.

Reset	Time to wait until the microcontroller circuit is reset/powered down
PowerUp	Time to wait for powering up the microcontroller circuit.
Transmit Delay	Delay to insert between transmitting two characters of the *Init String*. Only used if the device requires an initialization string.
Receive Delay	Delay to insert between receiving two characters of the *Response*. Only used if the device sends a response string.

Reinitialize every xx sec	If your receiver doesn't respond any more after some time, enable this to reinitialize the device every xx seconds. *This option may cause certain performance issues on some hardware configurations. You should leave it disabled if your device operates properly. If you need to turn it on, disable it when burning CDs or running similar applications.*

miroMEDIA Remote

Configuration

Select the COM port where the device is attached.

REALmagic Hollywood Plus Remote Control

Configuration

Select the COM port where the device is attached.

More Information

Visit REALMagic's website

Redlink 105 TV

Configuration

Select the COM port where the device is attached.

More Information

Visit the redlink website

Silitek SM-1000

Configuration

Select the COM port where the device is attached.

More Information

Visit Silitek's website

URCR 1.2

The URCR is a home-brew receiver device. With a bit of soldering experience, you can easily build it yourself. Visit the URCR website for circuit diagrams and PCB layouts.

Configuration

Select the COM port where the device is attached.

More Information

Visit the URCR website

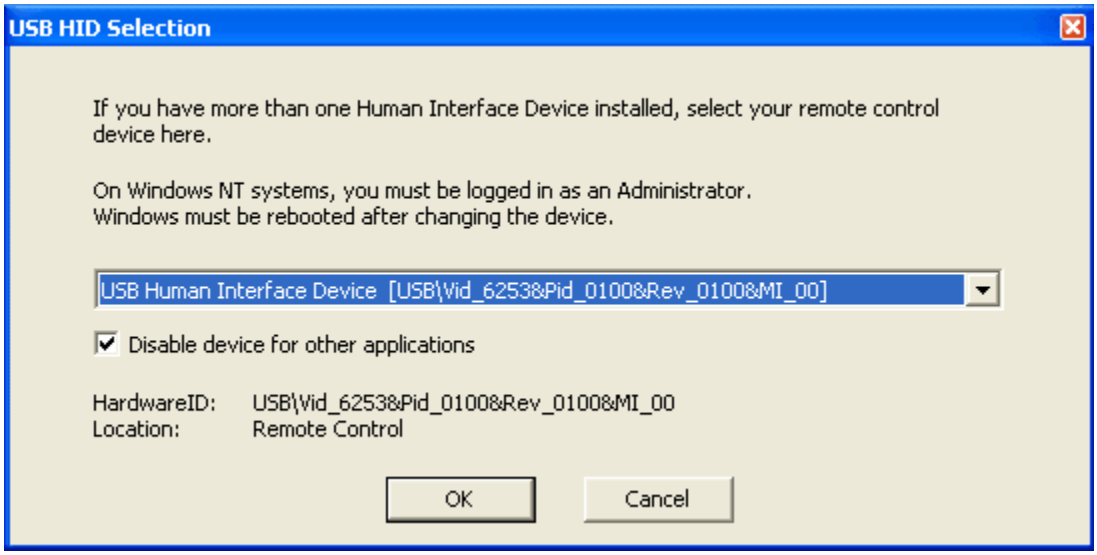
USB Human Interface Device

This is a generic driver for USB Human Interface Devices (HID) that appear in Windows Device Manager like the following:

HIDs can be any kind of input devices like remote controls, keyboards, game pads and mice, usually connected to one of the PC's USB ports.

Configuration

If you have more than one USB Human Interface Device installed in your system, click *Setup* to select your remote control device. The checkbox *Disable device for other applications* will disable any built-in functions of the device (e.g. mouse movement, keystrokes), allowing uICE to take full control over it.


USB HID Selection
If you have more than one Human Interface Device installed, select your remote control device here.
On Windows NT systems, you must be logged in as an Administrator.
Windows must be rebooted after changing the device.
USB Human Interface Device [USB\Vid_6253&Pid_0100&Rev_0100&MI_00]
Disable device for other applications
HardwareID: USB\Vid_6253&Pid_0100&Rev_0100&MI_00
Location: Remote Control
OK
Cancel

Other Devices

If you have another device not mentioned in the list, it is still possible that it works with one of the device plugins.

First, find out how your device is connected to the computer. If it is plugged into a serial port, you have a fair chance that it will work with either the Generic Serial IR Receiver or the Generic DCD Line Device. Currently two different technologies are supported, the first talks to the computer using the standard RS232 protocol (like a modem), the second uses the carrier detect line to send raw signals (pulses and spaces).

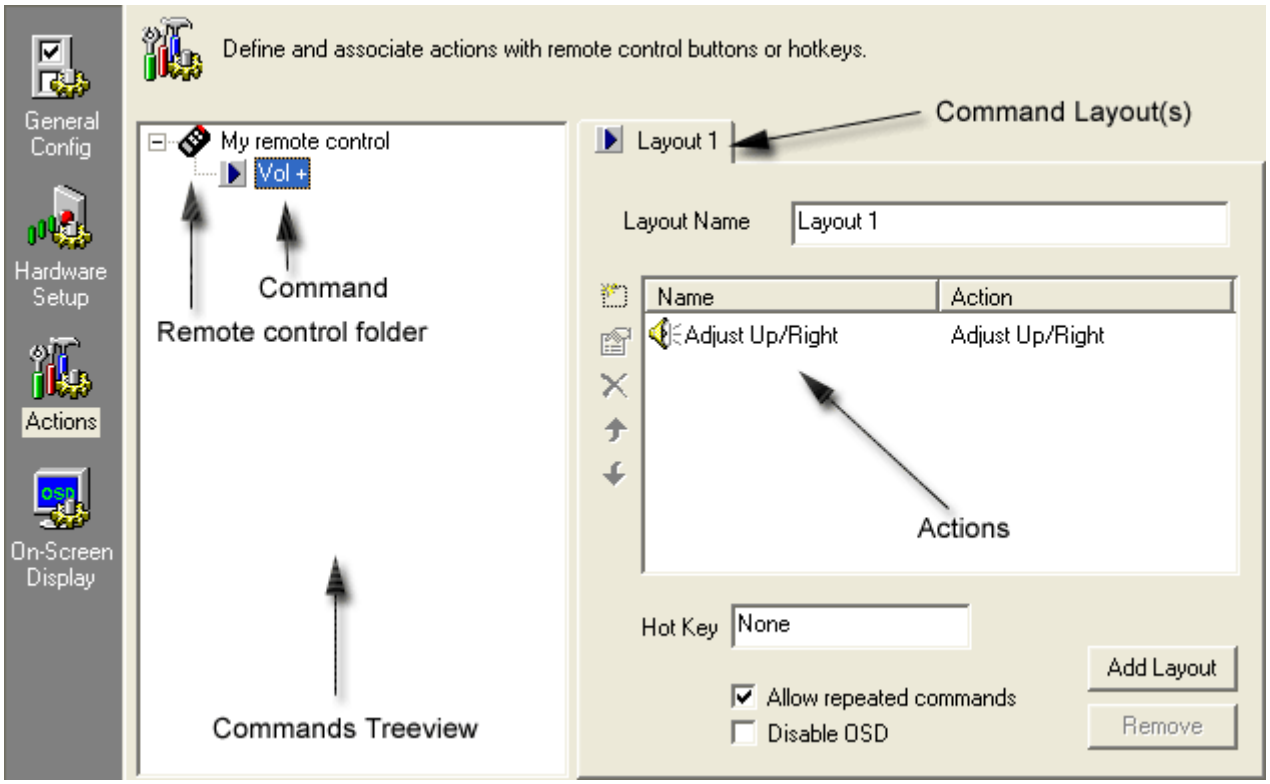
For the first, you will need to find out the proper settings for the COM port, i.e. baud rate etc. This is a trial-and-error procedure with the *Generic Serial IR Receiver* device.

The second requires no special configuration, so probably you should try this one first. Select the *Generic DCD Line Device*, make sure the correct COM port is selected and try to learn some buttons.

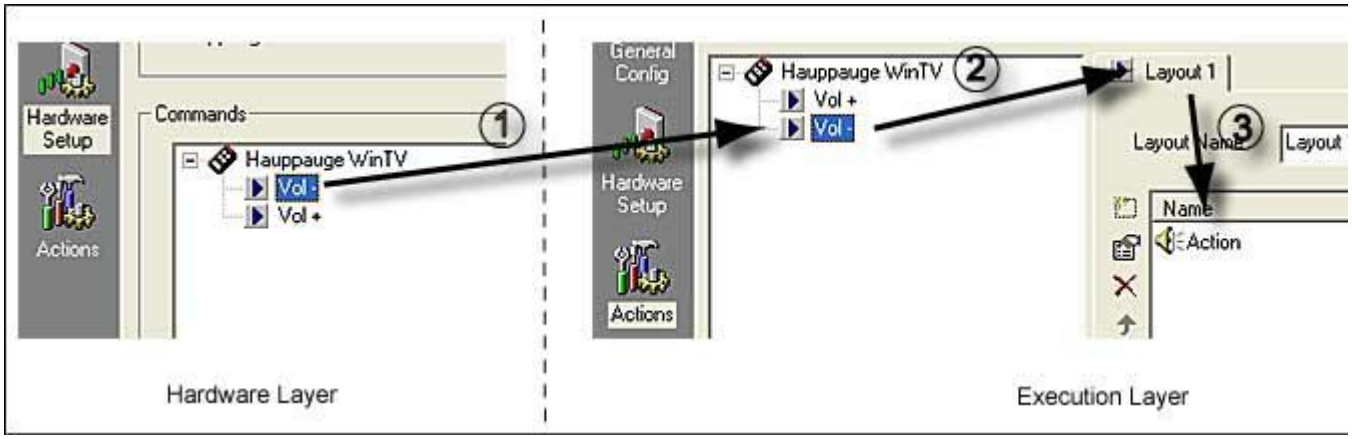
Additionally, if you have a BT8xx based TV tuner card with internal remote control receiver, you can try to use the *TV Capture Board* driver. Run the setup wizard for this device, then try to learn some buttons.

About Actions

To control applications and perform various system tasks, uICE provides a set of functions, called *Actions*. One or more actions can be combined into a layout and associated with a command.



The command, sent from the hardware layer in uICE when you press a button on your remote control, triggers the active layout and starts execution of the contained action(s). The following figure illustrates (simplified) the processing of input events from the remote control receiver from the viewpoint of the user interface:



1. The hardware layer received input from the hardware. After the raw data were translated (decoded) to a command, the hardware layer sends the command to the execution layer.

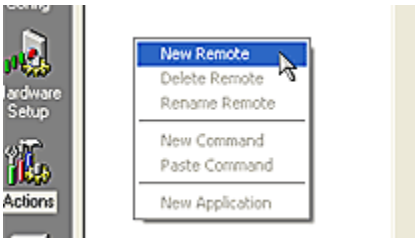
2. The execution layer locates the command and its associated layout(s) in the tree and determines the active layout (there's only one in this case).
3. The active layout executes its action(s).

The next topic shows how to define remote control folders, add and edit commands .

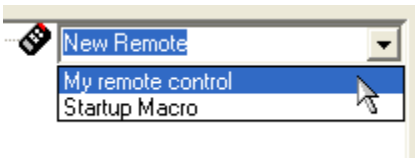
Adding and editing commands

The Actions treeview provides a context menu to add, delete and edit commands. Right click the treeview to open the menu.

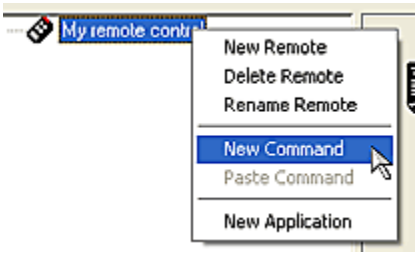
1. Add a remote control folder. From the context menu, select *New Remote*.



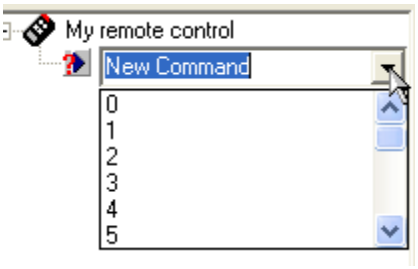
2. Give the folder the same name as in Hardware Setup. You can select it from the dropdown box, or just press a button on the remote control (if the hardware is already initialized).



3. Right click the remote control folder and select *New Command*.



4. Again, give the command the same name as one of the commands at Hardware Setup. The dropdown box lists the available commands, and you can also press the button on the remote control like in 2.

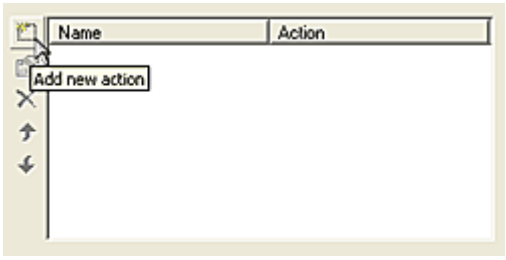


To rename or delete commands or remote control folders, select from the context menu or use the usual keyboard shortcuts like F2 or DEL. **All changes made at page Actions do not affect learned commands from Hardware Setup.**

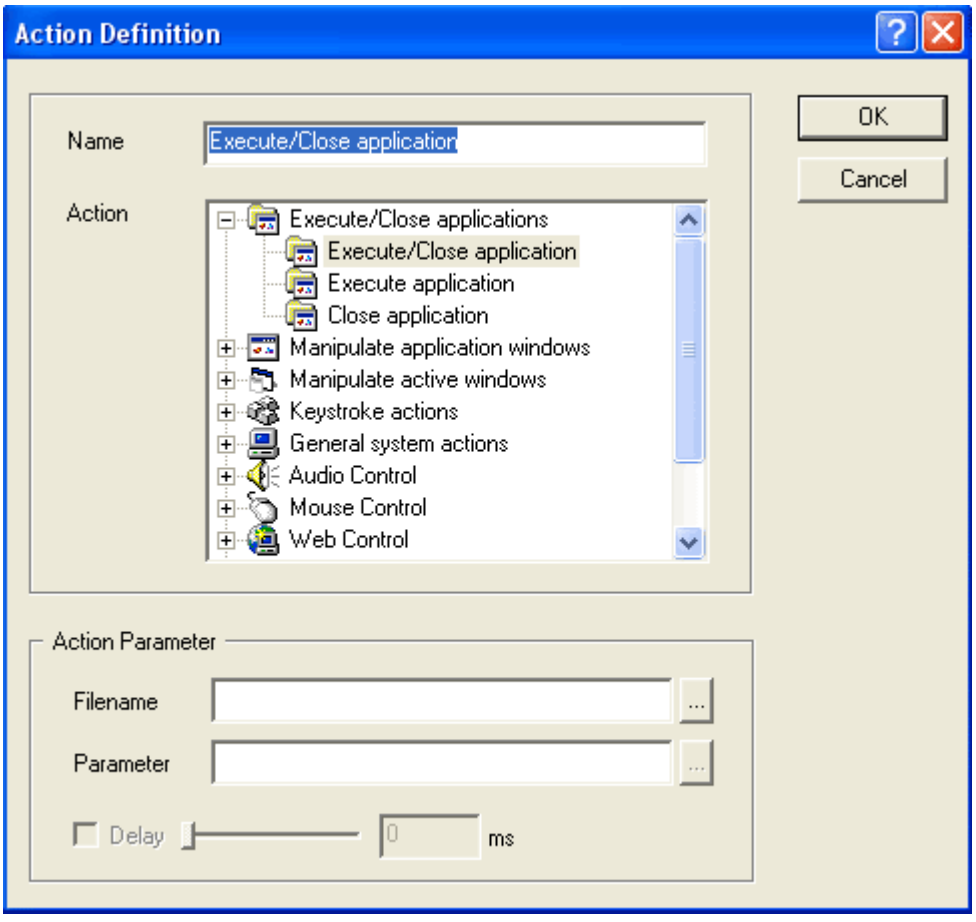
The following topic shows how to define actions .

Defining Actions

To add actions to a layout, use the toolbar left to the actions list or select from the context menu.



You will now see the *Action Definition* window:



Name

You can give the action a descriptive name. By default, uICE automatically fills in the label of the selected action.

Action

Select from the available actions. All actions and their parameters are explained in the Reference chapter.

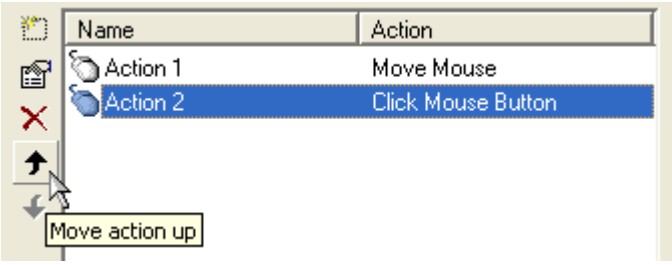
Action Parameter

Actions usually require one or more parameters. The buttons right the the parameter fields open assitants to easily define parameters.

The Delay checkbox, available for the second and all following actions in a layout, tells the execution engine to pause for an adjustable amount of milliseconds (ms) between the execution of two actions. If you apply a delay of 1000ms to the second action, the first action get executed, and then the execution will pause for one second before processing the second action.

Changing execution order

You can change the order in which multiple actions get executed using the up and down buttons from the toolbar.



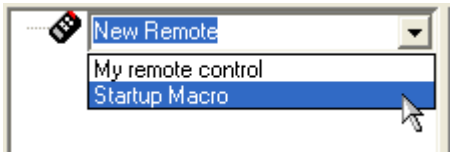
Editing actions

To edit an action, double click the action name or click the Properties icon in the toolbar. Cut&Paste operations are supported from the context menu to move or copy actions to a different layout or command.

The next topic explains the use of the Startup Macro, a special remote control folder.

The Startup Macro

uICE supports a special, predefined remote control folder, the *Startup Macro*. You can add one single-layout command to this folder, with as much actions as you like. All actions of this command get executed immediately after starting uICE.



The next chapters introduce two important parameters used in several actions:

Window Paths, to reference a certain window on the desktop, and

Window Messages, Microsoft Windows' basic communication architecture.

Window Paths

Several actions operate on certain desktop windows, like PostMessage or Keystroke to window. The Operating System identifies windows on the desktop using a unique numerical value, called a *window handle*. These handles are only valid for the life cycle of a window (i.e. the same window gets a different handle if you close and reopen it). To be able to identify a window even between two sessions, uICE uses a different approach, called *window paths* .

Windows are organized into a tree structure (see **Fig.1**), and grouped by the executable's name which created the window. All windows directly beneath an executable (e.g. *about:blank - Microsoft Internet Explorer (IEFrame)* beneath *IEXPLORE.EXE*) are so called top-level windows. Top-level windows have a caption bar and usually a system menu, and they can create child windows like buttons, toolbars or other windows without title bar to display documents for example. Every standard Windows application with user interface has one or more top-level windows, one of these windows is usually called the application's main window.

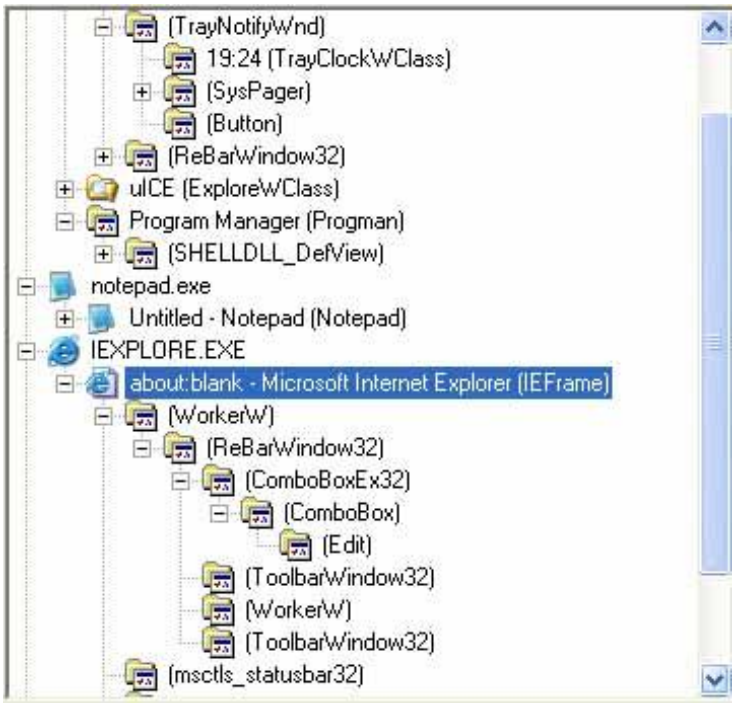


Fig.1

To identify a top-level window, uICE uses the executable's name, the title (the name that appears in the caption bar) and classname of the window. The classname belongs to an internal Windows structure describing a certain class of windows, for example dialog boxes. Neither the title nor the classname need to be unique across the system, but most windows can be identified by these two names. The Internet Explorer's main window, as highlighted in the window picker tree image above, with the title *about:blank - Microsoft Internet Explorer* and the classname *IEFrame* would be identified by its window path *IEXPLORE.EXE\about:blank - Microsoft Internet Explorer (IEFrame)*. Note the backslash delimiter, which separates a window from its childs and the executable, similar to files on your harddisk. To identify a child window of IE's main window, let's say the window with the classname *ReBarWindow32*, we would walk the tree down to this window and append each child window: *IEXPLORE.EXE\about:blank - Microsoft Internet Explorer (IEFrame)\(WorkerW)\(ReBarWindow32)*

Note that windows can change their title at any time. Once you use Internet Explorer to navigate to a website, the title of the main window changes to the title of the website, as shown in **Fig.2** :

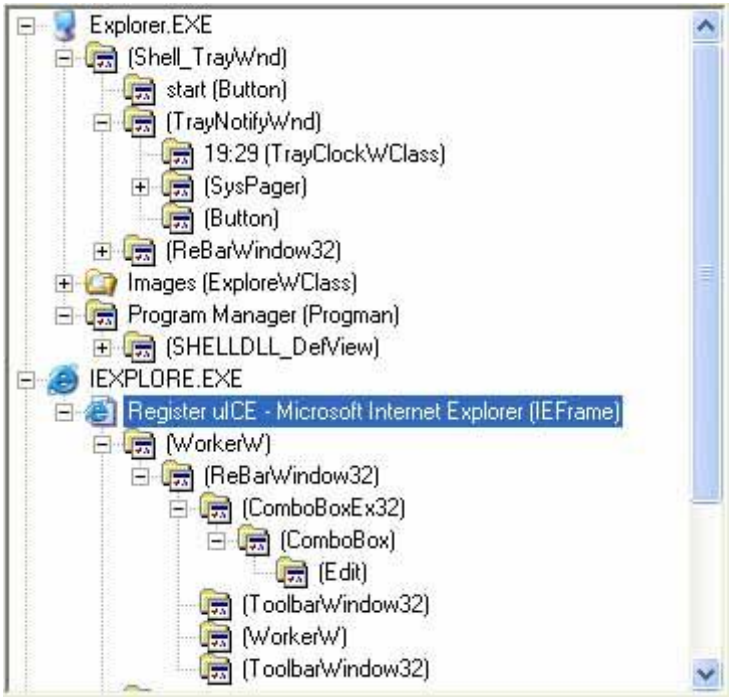


Fig.2

Now the window path would change to *IEXPLORE.EXE\Register uICE - Microsoft Internet Explorer (IEFrame)*. For applications that change their window titles at runtime, uICE supports wildcards (*,?). You can substitute the changing part of the window path, like in IE's case: *IEXPLORE.EXE* - Microsoft Internet Explorer (IEFrame)*. Now IE's main window can be identified regardless of the website's title.

Note: There are still certain limitations in window paths. If more than one window with the same window name and class name belongs to the same executable, uICE may not be able to determine the correct window. Instead, the first window in the tree matching these names will be chosen. Press Mark in the list of available windows to see which window uICE has found.

Introduction to window messages

This chapter will show you a bit of the backgrounds behind Microsoft Windows' basic communication mechanism: *window messages*.

Windows based applications are event driven, i.e. they get notifications and informations from the Operating System about occured events (e.g. user input events like typing, mouse movements and mouse clicks, selection of menu items or button clicks) in form of a special system structure, called a *message*. In the other direction, an application can generate a message to the system (or other applications) to perform certain tasks or to inform about changes.

A message consists of four parameters: a *window handle*, a *message identifier* and two values called *message parameters*. All four parameters are numerical values.

The *window handle* uniquely identifies the window across the system for which the message is intended. Since a window handle is only valid for the life cycle of a window (i.e. the same window gets a different handle if you close and reopen it), uICE uses a different approach to identify a window between two sessions: window paths.

Message identifiers are unique, predefined values, usually represented by aliases from the Win32 Software Development Kit. Most system messages are meaningless in regards of application remote control, so the number of interesting messages is reduced to a reasonable amount. These messages will be explained further in Message types .

While the message identifier represents the system notification about the kind of occured event, the two *message parameters* give further information about the context of the event. As an example, when the user hits a key on the keyboard, the message identifier notifies the application window that *one of the keys* has been pressed. The window which receives this notification can now examine the message parameters to determine *which* key it was.

Now why is this so important in regards of application remote control? As said, the Operating System notifies applications about user input. uICE is able to monitor the communications between the system and other applications, record user input events and replay them on demand. With this feature, you are able to simulate keystrokes, menu item selections, pushbutton clicks and mouse clicks. Since messages are Windows' native communcation mechanism, it is recommended to always try to control an application using this method. It is the fastest and safest way.

One of the Win32 API functions which allow an application to generate a message is called PostMessage. This is the function uICE uses to replay previously recorded messages. The following topics will show you how to record messages and explain the different types of messages and their meaning.

Recording messages

This topic shows a step-by-step example on how to record a message representing a menu selection. We will use Microsoft Internet Explorer (version 5 or 6) as the example application, and set up a PostMessage action to open and close Internet Explorer's Favorites toolbar. If you are not yet familiar with Windows' message mechanism, read the Introduction to window messages first.

- First, start uICE and Internet Explorer
- Go to page Actions, create a new command and add a new action to the command. The *Action Definition* window will pop up.
- Now select the PostMessage action, located under *Window Messages* .
- In the Action Parameter section of the window, click the button right to the Window field (make sure Internet Explorer is running)

- uICE now shows the running applications on their windows organized in a tree. Find Internet Explorer's executable (*IEXPLORE.EXE*) and select it.
  If you use the autoselect feature (i.e. moving the mouse cursor over the window and hitting F2 on the keyboard), uICE will automatically select the window beneath the mouse cursor, which is usually one of the main window's child windows. For this example, select the executable to let uICE monitor Internet Explorer's complete message traffic.
- Click OK to apply the selection. Now click the button right to the Message field. You'll see the *Message Capture* window, which records the user input messages sent by the system to Internet Explorer's main window.
- From Internet Explorer's main menu, select *View|Eplorer Bar|Favorites*. This opens the Favorites toolbar.
- Go back to the Message Capture window. The list now shows the captured messages. You should see several left mouse clicks (WM_LBUTTONDOWN) and one button/menu command (WM_COMMAND). The WM_COMMAND message represents the menu selection. Select this message and click OK.
- The Message field now shows the recorded message, which should look like this (for IE6): *WM_COMMAND:0x0000a232:0x00000000*. The format of this string is *message_alias:message_parameter_1:message_parameter_2*.
- Click OK in the Action Definition window. The PostMessage action is now ready to use.

To test the action, right click the command's label and select *Test* from the context menu. You'll see Internet Explorer's Favorites toolbar open and close.

Essential for capturing and proper operation of messages is the selection of the correct window. If you are not able to capture messages using the application's main window, try to capture from one of its child windows.

The next topic will explain the different types of capturable messages.

Message types

This topic describes the different types of messages used for application remote control. If you are not yet familiar with Windows' message mechanism, read the Introduction to window messages first.

The format used in the Message field of the Action Definition window is
message_alias:message_parameter_1:message_parameter_2

WM_COMMAND

The system sends a WM_COMMAND message to an application when the user selects a menu item (either from the main menu or a context menu) or clicks a pushbutton in a dialog window or toolbar. If the application has a menu, most of its functions can be simulated with this message.

WM_KEYDOWN / WM_CHAR

WM_KEYDOWN and WM_CHAR messages represent a keystroke on the keyboard. The basic difference between the two messages is, that a WM_KEYDOWN message gets translated by the system to a hardware independent WM_CHAR message.

The advantage of using keystroke messages over one of the keystroke actions is, that uICE does not need to change the window order on the desktop, which saves processing time.

Each application is responsible to handle keystroke messages properly. Some applications however monitor the keyboard hardware itself, which can't be simulated with messages. For these applications, you'll need to use one of the Keystroke actions.

WM_LBUTTONDOWN / WM_LBUTTONDBLCLK

Generated by the system when the user clicks (WM_LBUTTONDOWN) or double-clicks (WM_LBUTTONDBLCLK) the left mouse button in a window. The message parameters contain the coordinates of the mouse cursor relative to the window beneath it, the state of the SHIFT and CTRL key and the other mouse buttons.

Execute/Close applications

To select a file, click the button right to the Filename field. Environment variables in the form `%WINDIR%` are supported for file system paths.

Tip: It is always a good idea to close applications using the PostMessage action instead of Close application. Several applications have their own procedure to shut itself down and release resources and hardware they might be using, while Close application sends a generic Windows command to the program that it should close. The application itself is responsible to terminate correctly in response of this command. Most do it, some however not. With PostMessage, you can simulate the selection of the *Exit* or *Quit* item from the application's main menu (*File*|*Exit* or *File*|*Close*) just like you were closing it by hand, which of course shouldn't confuse even applications that do not handle the generic command correctly.

Execute application

Executes an application.

Field	Required ?	Value
Filename	Yes	Full path and filename of application to start
Parameter	No	Optional command line

Close application

Closes an application.

Field	Required ?	Value
Filename	Yes	Filename of application to close. A path is not required.

Execute/close application

Executes an application if it's not running, closes it otherwise

Field	Required ?	Value
Filename	Yes	Full path and filename of application to start/close
Parameter	No	Optional command line

Manipulate active windows

Minimize current window

Minimize the window currently in front of the screen.

No parameters.

Maximize current window

Maximize the window currently in front of the screen. The next button push will restore the window to its normal size.

No parameters.

Close current window

Close the window currently in front of the screen.

No parameters.

Switch to window

Puts the specified window into foreground.

Field	Required ?	Value
Window	Yes	Path to window. Press the button to select from currently open windows.

Cycle through windows

Opens the Windows Task Manager. The first button opens the Task Manager, any subsequent push cycles through the open windows/applications. When you have found the desired window, push any other button with an assigned command to activate the selected window, or just wait a second until the Task Manager disappears.

No parameters.

Manipulate application windows

To select a file, click the button right to the Filename field.

uICE manipulates the application's main window. Since many applications create more than one top-level window and there's no way to determine the main window from the application's file name, uICE needs to "guess" the correct window. This may fail for certain applications.

Min/Max/Restore application

Minimize, maximize and restore an application's main window with consecutive button pushes.

Field	Required ?	Value

Filename	Yes	Filename of application. A path is not required.

Minimize application

Minimize an application's main window. The next button push will restore the window to its normal size.

Field	Required ?	Value
Filename	Yes	Filename of application to minimize. A path is not required.

Maximize application

Maximize an application's main window. The next button push will restore the window to its normal size.

Field	Required ?	Value
Filename	Yes	Filename of application to maximize. A path is not required.

Restore application

Restore an application's main window to its normal size.

Field	Required ?	Value
Filename	Yes	Filename of application to restore. A path is not required.

Keystroke actions

The basic keystroke actions (Keystroke to application, Keystroke to window) require a window (or application) to be put into foreground before it can accept a keystroke. uICE uses a mechanism to restore the window order on the desktop to its previous state a few seconds after the keystroke.
You may disable this mechanism by enabling Leave window focus after keystroke, or use PostMessage instead, which is able to simulate keystroke messages without changing the window order on the desktop.

Keystroke to application

Send keystroke to an application

Field	Required ?	Value
Filename	Yes	Filename of application. A path is not required.
Keystroke	Yes	Keystroke definition. Click to open the keystroke helper.

uICE tries to send keystrokes to the application's main window. If the application's main window was not created by its executable file image, uICE cannot determine the proper window from the file name. Many applications also create more than one top-level window, so uICE needs to "guess" the correct window. This may fail for certain applications.

Keystroke to current window

Send keystroke to the window currently in front of the screen

Field	Required ?	Value
Keystroke	Yes	Keystroke definition. Click to open the keystroke helper.

Keystroke to window

Send keystroke to a specific window

Field	Required ?	Value
Window	Yes	Path to window. Click button to select from currently open windows.
Keystroke	Yes	Keystroke definition. Click to open the keystroke helper.

The Keystroke Helper dialog

This dialog lists the available keys in uICE. Select a key in the list or type a key in the edit box, check the ALT, CTRL and/or SHIFT options that shall go along with the key and press Insert. Right click a key in the list to deselect it. When finished inserting keys press Done.

The dialog also provides an Auto Key Capture feature. Left click the capture field an press the desired key or key combination. The keystroke will be automatically inserted into the Keystroke field, pressing Insert is not required.

Keystroke Syntax

- To send a normal readable (ASCII) character, type the character into the field. You can type whole sentences, not only single characters.
- Specify special keys, like TAB or PGUP, by using an identification number, enclosed in brackets { }. See the file vk.key located in the uICE directory for those numbers. To type a hexadecimal value as given in the vk.key file,

the number has to start with "0x" (without the quotation marks).

- To simulate the ALT, SHIFT or CTRL key along with another key, type the leading character (A, S or C) of the key enclosed in brackets { }, followed by the key that shall be pressed with one of those keys. The ALT, SHIFT and CTRL statement is only valid for the next following key.
- It is not necessary to use the {S} statement to get readable upper case characters like 'A'. {S}a is the same as just A.

Examples:

a means a
{S}a means A
{S}hello means Hello
u{S}I{S}C{S}E means uICE (equal to just typing 'uICE')
{C}a means CTRL-a
{CA}a means CTRL-ALT-a
{SCA}a means CTRL-ALT-A
{0x70} means F1 key
{C0x70} means CTRL-F1

General system actions

Shutdown Windows / Reboot Windows

Shuts down or reboots your system. For shutdown, the power is turned off if your hardware supports this function (ATX power supply).

By default, uICE displays an OSD countdown of 15 seconds. During this countdown, you may abort the process. If you prefer to shutdown/reboot the system immediately without countdown, disable the OSD for the command that triggers this action.

The sleep timer known from previous versions of uICE can now be found in the On-Screen Menu.

No parameters.

If you want to ensure that no other application prevents the system from shutting down, check Enable forced system shutdown at General Options.

Suspend Windows

Suspends your system. Timed suspend is available in the On-Screen Menu.

Field	Required ?	Value
How	Yes	Select from two suspend methods: Standby or Hibernate. Depending on your computer's capabilities and BIOS settings, Standby may switch your system into Suspend-To-RAM mode. Windows NT based systems (2000/XP) require ACPI compliant hardware. Hibernate is not supported on Windows 98 systems.

Before using this action, test your system by saving all open documents and putting it into standby mode manually. Many badly written device drivers may prevent the system from waking up again properly.

Turn off display

Switches the display into energy saving mode. Multi-monitor configurations are not supported.

No parameters.

Change Display Mode

Change the current display mode. The dropdown box lists the available modes.

Field	Required ?	Value
Mode	Yes	Select the desired display resolution from the dropdown list.

Audio control

Audio Mode

Cycles through sound controls. Starting with Volume Control, any subsequent button push within two seconds will call adjustment mode for the following controls (if available for the soundcard and selected input line): Pan, Bass, Treble, Fader. To adjust the current mode, use Adjust Up/Right and Adjust Down/Left. This function is similar to those seen in many car stereos.

Field	Required ?	Value
Mixer	Yes	Select the output line from the dropdown list

Adjust Up/Right

Increases the value of the current audio mode. If the last Audio Mode command has timed out (after two seconds), the volume of the selected mixer input line will be increased.

Field	Required ?	Value
Mixer	Yes	Select the output line from the dropdown list
Accel	No	Acceleration level (1..20). Adjust this value to let the volume increase more ore less faster. If not applied, a default acceleration level is used.

Adjust Down/Left

Decrease the value of the current audio mode If the last Audio Mode command has timed out (after two seconds), the volume of the selected mixer control line will be decreased.

Field	Required ?	Value
Mixer	Yes	Select the output line from the dropdown list
Accel	No	Acceleration level (1..20). Adjust this value to let the volume decrease more ore less faster. If not applied, a default acceleration level is used.

Mute

Mutes / Unmutes a mixer line.

Field	Required ?	Value
Mixer	Yes	Select the output line from the dropdown list

Record Volume

Increase or decrease the volume of a record mixer line.

Field	Required ?	Value
Mixer	Yes	Select the input line from the dropdown list
Accel	No	Acceleration level (-20..20). Adjust this value to let the volume increase (positive values) or decrease (negative values) more or less faster. If not applied, a default positive acceleration level is used.

Mouse control

Move Mouse

Moves the mouse cursor.

Field	Required ?	Value
Direction	Yes	Movement direction (North, East, ...)
Accel	No	The acceleration level (1..10). Adjust this value to let the mouse cursor move more or less faster. The default is 6.

Set Mouse Position

Sets the position of the mouse cursor.

Field	Required ?	Value
Relative To	No	Set the position relative to the Desktop (if field is blank) or a

		specific window. Click the button to open the position wizard.
Position	Yes	Position coordinates. These are logical coordinates (0..65535) to be independent of screen resolution. Use the position wizard to easily apply the correct coordinates.

Roll Mouse Wheel

Rolls the mouse wheel.

Field	Required ?	Value
Direction	Yes	Roll direction (Up/Down)
Accel	No	The acceleration level (1..255). Adjust this value to let the wheel roll more or less faster.

Click Mouse Button

Clicks a mouse button.

Field	Required ?	Value
Button	Yes	The mouse button to click (Left/Middle/Right)

DoubleClick Mouse Button

Double-clicks a mouse button.

Field	Required ?	Value
Button	Yes	The mouse button to click (Left/Middle/Right)

Web control

Connect

Connects to your ISP using Dial-Up networking.

Field	Required ?	Value
Connection	Yes	Select a phonebook entry. User name and password need to be saved from Windows' Connect dialog.

Disconnect

Disconnects any active dial-up connections.

No parameters.

Open Location

Opens a web location in the default browser.

Field	Required ?	Value
Location	Yes	The location's URL (e.g. http://www.mediatexx.com)

Window messages

PostMessage

uICE can capture and re-send messages sent to application windows. This feature is able to simulate the selection of menu items, push-button clicks, mouse (double-)clicks and keystrokes without changing the window order on the desktop.

Read the Introduction to window messages to learn more about this very important feature.

Field	Required ?	Value
Window	Yes	Specify the target application's window path. Click the button to see a list of currently open windows.
Message	Yes	Press the button to open the message capture window.

Application Command

Sends an application command to a window. Application commands (also known as WM_APPCOMMAND) have the same functionality like pressing a multimedia key on a multimedia keyboard.

Field	Required ?	Value
Window	No	Optionally specify the target application's window path. Click the button to see a list of currently open windows. If empty, the application command goes to the window currently in the foreground.
Command	Yes	Select from available application commands.

Engine actions

Switch Layout

Switch command into another layout. If Switch all command layouts simultaneously is enabled for application presets, this will switch all other commands into the corresponding layout as well, e.g. switching to the second layout switches all other commands of the same application preset to the second layout (regardless of layout names).

Field	Required ?	Value
Layout	Yes	Select from available layouts.

Set active application

Set the active application preset. Provides the same functionality as the Applications submenu in the On-Screen Menu. Use this function as a shortcut to execute/switch to an application and set its commands active.

Field	Required ?	Value
Application	Yes	Select from available application presets

Load configuration

Loads one of the saved configurations you can also load manually from *Preferences*|*Configurations*.

Field	Required ?	Value
Configuration	Yes	Select from available configurations

RunScript

Runs a script. Click the button to open the script editor. See About Scripts for an introduction.

OSD actions

Display window title

Displays the title of other windows in an On-Screen display. The contents of static controls, edit boxes and combo boxes can be displayed as well.

Field	Required ?	Value
Window	Yes	Path to window . Click the button to select from available windows.

On-Screen Menu shortcut

Opens the On-Screen Menu and goes directly to the selected menu page.

Field	Required ?	Value

Menu Item	Yes	Select the menu page to go to.

Overview

The amount of remote control buttons is usually quite limited. Common remote controls have only 20-30 buttons, which is just not enough to control the numerous functions of today's software technology. To overcome these limitations, uICE provides two unique features that allow assignment of different functions to a single remote control button.

Application-specific Commands

You can organize buttons/commands in application presets. The commands contained in such a preset are only active if the application itself is active. This allows you to define completely different command sets for each application.

Multi-Layout Commands

Even within the same application, it may be required to have different actions assigned to a single command. Multiple layouts allow assignment of different actions to a single command.

Additionally, you do not need to waste buttons for switching between applications or layouts. Both functions are available through the On-Screen Menu.

Application Presets

Application presets contain command sets with actions specific for an application. These commands get executed only if the respective application is running and active .

To add an application preset, right click the remote control label in the tree and select *New Application* from the context menu. You can now type a descriptive name for the application (this name will be used for On-Screen Menu items), or select one of the preconfigured applications from the dropdown box.

When using application presets, you can still define global commands :

- After a command has been received from the remote control, uICE looks for active applications first. If one was found and the received command matches one of the application commands, uICE executes it and disregards any existing global commands with the same name.
- If no command within the active application preset matches the received one, uICE looks for a matching global command and executes it if found.
- If none of the configured application is running, uICE executes a matching global command.

Application presets may be exported to make them available in the *New Application* dropdown box. Right click the application preset's label and select *Export Application* from the context menu. The application commands will be saved to the file `<application name>.uap` in the `uICE\apps` directory on your harddisk.

Updated and new application command files can be found at the uICE website . To install a downloaded `.uap` file, you just need to copy it into the `uICE\apps` directory. It is then available in the *New Application* dropdown box. **After importing a downloaded preset, remember to rename the application commands to the names of your remote control** .

Filename

The application's full path and file name. This parameter is required.

Execute command

You can assign a quick-access command to execute the application. This is optional, you can also execute and activate applications through the On-Screen Menu.

The *Advanced* button lets you customize the action(s) assigned to this command. By default, uICE sets an Execute application action, but you can for example add another action to automatically maximize the main window after starting.

The action(s) get also executed if the application is not running and you select it through the On-Screen Menu.

Main Window

You can optionally specify the application's main window here. uICE will then use it as the default window when configuring actions that require a window path. Click the ... button to select the window.

Disable remote control for this application

If checked, all remote control commands except the Execute command are disabled if this application is active.

Switch all command layouts simultaneously

By default, switching a command into another layout does not affect the layouts of other commands.

If this option is enabled, switching the layout of one command will cause all other commands contained in this

preset to be switched into a corresponding layout.

Example: if you switch button '0' into the second layout, all other commands will also be switched into their second layout (regardless of layout names).

This option is useful to define complete command sets for different application modes that require different actions for each mode, such as viewing and browsing pictures.

Remember last active layout

If this option is enabled, currently active command layouts are saved when exiting uICE and restored when restarting.

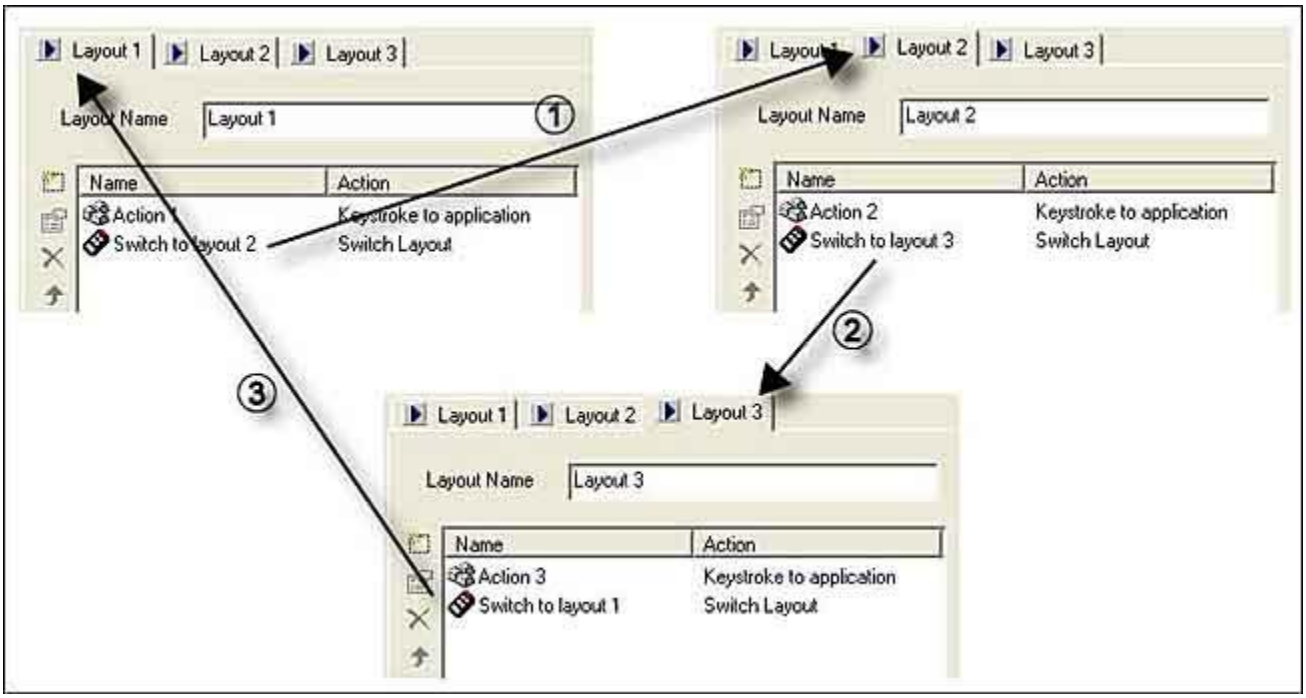
Multi-Layout Commands

A layout is a container for one or more actions. The actions contained in a layout get executed one after the other whenever a command was received from the remote control.

Multi-Layout Commands offer the ability to assign multiple action sets to a single command. Each layout is independent from other layouts, only the actions contained in the currently active layout get executed.

Setting the currently active layout can either be done manually using the Layout Selector from the On-Screen Menu, or automatically after the associated command has been received by adding a Switch Layout action to the command's layouts.

Automatic layout switching is useful to define toggling or even multi-state commands. As an example, think of a command with three layouts. Each layout has one action to carry out a specific application function, and an additional Switch Layout action to set the active layout to the next one automatically. The third layout's Switch Layout action sets to active layout to the first one again. The result is a tri-state command:



Layouts can also be used to provide action sets for different application modes. The preconfigured application preset for Internet Explorer makes use of this functionality:

This pictures shows the Favourites command, which toggles Internet Explorer's Favourites toolbar.

IE requires different actions to navigate through the favourite shortcuts and browse a webpage location. The Favourites command switches the complete application preset between the two modes:

- The first time the Favourites command gets executed, IE's Favourites toolbar opens and the Switch Layout action instructs uICE to switch all commands of the preset to their second layout. For this to work, Switch all command layouts simultaneously must be enabled for the application preset, and all commands within the preset must have two layouts.

- The second time the Favourites command gets executed, the toolbar closes and the Switch Layout action within the *Browse Location* layout switches all commands back to their first layout.

About OSDs and the On-Screen Menu

uICE provides On-Screen Displays (OSD) and an On-Screen Menu (OSM) similar to those known from TV sets. By default, every executed command displays an OSD, either with special information like for volume control or containing the button name and the actual layout name. OSD windows can be disabled for certain actions by checking the Disable OSD box in the command definition.

The On-Screen Menu features special functions to control multiple applications at the same time, to switch between command layouts, and timers to shutdown the computer after a certain time (Sleep-Timer) or wake it up from standby mode (Wakeup-Timer).

The next topic explains the functionality of the On-Screen Menu.

On-Screen Menu items

Applications

The *Applications* submenu lists the configured application presets. When selecting one of the applications, uICE executes it (if it's not yet running) or brings its main window to the front. Once the application is selected, the commands of its preset become active.

To control another application, select it from the *Applications* submenu. The commands of the previously active application get disabled, and the commands of the selected application become active. This way allows you to control multiple applications at the same time.

Layout Selector

The *Layouts* submenu lets you manually switch multi-layout commands.

After selecting *Layouts* , press a multi-layout button on your remote control. You can now select from the available layouts.

Files

With the *Files* submenu, you can browse and launch files from your harddisk. Four filter modes are available: *Videos*, *Pictures*, *Music* and *All Files*. uICE remembers the last used folder for every filter, so if you organize your video/picture/music files into different folders, you have fast access to them.

When selecting a file, uICE launches its associated application (like Windows Media Player for video files). Directories on your harddisk are enclosed in square brackets []. *Videos* and *Music* have a *[Play all from xxx]* item, which creates an ASX or M3U playlist and launches the associated application with it.

You can customize the application being launched in Windows' file association properties (Control Panel->Folder Options->File Types).

Timers

The *Timers* submenu features two items: a Sleep timer and a Wake-up timer.

Before using the timers, test your system by saving all open documents and putting it into standby mode manually. Many badly written device drivers may prevent the system from waking up again properly.

The Sleep timer turns your computer off after an adjustable amout of time, up to 99 minutes. You can additionally select how uICE should turn it off: Standby, Hibernate or Shutdown. Some of this methods may not be supported on your system. Use the Left/Right menu commands or the number buttons on your remote control to adjust the time.

The Wake-up timer wakes your computer from Standby or Hibernate mode at the specified time. This functionality is only supported on ACPI compatible computers and Operating Systems like Windows 2000 and Windows XP. Use the number buttons on your remote control to adjust the time.

The next topic explains how to navigate the On-Screen Menu.

Menu Commands

To navigate between On-Screen Menu (OSM) items, uICE uses the commands from Hardware Setup. The *Menu Commands* page lets you assign commands to menu navigation actions like up, down, left and right. Only the *Menu/Back* command is reserved to bring up the OSM, all other commands are still available for use at Actions and only active if the OSM is visible.

Menu Action	Description	Keyboard control
Menu/Back	Enters On-Screen Menu mode. If the On-Screen Menu is active, goes one level up in the menu hierarchy or turns it off if the main menu is displayed.	- OSM not active: Configured Hotkey command from Hardware Setup - OSM active: ESC, Backspace or Hotkey command
Enter/Select	Selects the currently highlighted item from the On-Screen Menu.	Enter or Hotkey command
Up	Moves the On-Screen Menu selection marker up.	Cursor Up or Hotkey command
Down	Moves the On-Screen Menu selection marker down.	Cursor Down or Hotkey command
Left	Switches the currently selected On-Screen Menu item, for example the way the sleep timer suspends your computer. In Filebrowser mode, performs a page up.	Cursor Left or Hotkey command
Right	Switches the currently selected On-Screen Menu item, for example the way the sleep timer suspends your computer. In Filebrowser mode, performs a page down.	Cursor Right or Hotkey command
1..0	Used to enter digits in the On-Screen Menu's Sleep and Wake up timers.	1..0

The next topic shows how to customize the On-Screen Menu.

Visual Style

The Visual Style page lets you customize colors, font and background for the On-Screen Menu (OSM) as well as position and display duration for other On-Screen Displays (OSD).

Transparency

Set the transparency of OSDs and OSM. Transparent/translucent backgrounds are only available in Windows 2000 or higher.

Background

Set the background color of OSDs and OSM.

Selection Mark

Customize the color of the OSM's selection mark. Click the colored field to select a color.

Text

Customize color and font of OSD/OSM texts. Click the colored field to select from the available fonts.

Use Double-Buffering

Enable this checkbox to use double buffering technology to display the OSM. Double-buffering prevents flickering, but requires more processing power. If the OSM responds slowly, disable this option.

Menu Background Image

Enable this checkbox to use a background image for the OSM instead of solid colors.

Background Fading Effect

If enabled, the OSM's background is faded in when opening the menu.

Timeout

Adjust how long, in seconds, OSDs are to be displayed.

Border

Adjust the screen border for OSDs. Very useful for TV output with overscan.

Position

Select the screen position where OSDs are to be displayed.

About Scripts

uICE provides full support for Active Scripting, Windows' built-in, language independent scripting features. Similar to creating dynamic web pages with Javascript/JScript or VBScript, you can let commands execute script operations to combine uICE actions with the raw power of a full programming language - without the need of a compiler. To run a script, use the Run Script action under Engine actions.

Since Active Scripting is completely language independent, you can either use the Operating System supplied JScript/VBScript languages, or choose from a variety of third-party script engines like LUAScript, PerlScript, PythonScript and many others. If you've ever written a script for a web page, you can almost instantly program uICE.

Similar to Internet Explorer's window/document object model, uICE exposes fully featured, object-oriented access to all available actions, and also a few additional goodies like timers, global variables, programmable OSDs and access to the Distributed COM (DCOM) architecture.

Getting Started

Since every programming introduction seems to need its "Hello World"-Firststep-Example, here we go:

Create a Run Script action (under the Engine Actions category).

Open the script editor (click the button). Select JScript from the Script Language dropdown box.

Type the following line of code into the edit box.

```
Alert("Hello World");
```

Now click the toolbar's Run icon ( ).

Note that you can simply type some statements, without declaring functions or procedures. If you prefer to have a main function, or want to return an exit code (see Exit method for meaningful values):

```
function main()
{
  Alert("Hello World");
  return 0;
}
```

does the same.

In VBScript, it looks like this:

```
Alert "Hello World"
```

or

```
Function main()
  Alert "Hello World"
  main=0
End Function
```

A slightly more advanced example (JScript):

```
wmp6=CreateObject("MediaPlayer.MediaPlayer.1");
//note: the following statement assumes that you have a file
//"tada.wav" in your windows\media folder (usually you do)
```

```
wmp6.Filename=Env("windir")+"\\media\\tada.wav";
wmp6.Play();
```

The following sections describe the uICE object model, assuming that you're reasonably familiar with writing scripts. For tutorials and reference material on scripting languages, visit the MSDN library at msdn.microsoft.com or use one of the innumerous resources available on the web.

Let's start with the top-level object, called - guess what - uICE.

uICE Object

The uICE object is a global namespace object to access the functional objects, methods and properties provided by uICE (similar to the window and document objects in Internet Explorer). Since it is global, you don't need to prefix any exposed objects or methods with "uICE" in general. JScript example: `uICE.Alert("Hello World");` is the same as `Alert("Hello World");`

However, in the case of naming conflicts with script language built-in objects or methods, you can use the uICE object to tell the script interpreter which method is to be used. An example is the CreateObject function in VBScript, which is also a method of the uICE object. To use the VBScript function, simply type CreateObject, to use the uICE method, type uICE.CreateObject.

Each script runs in its own environment and is deleted from memory after it has finished execution (unless you call the SetTimer method). This implies that any variables used in a script are destroyed. If you need to save variables between two executions, or to pass variables between different scripts, use the Globals object.

Methods

Alert	Displays a text-only OSD
CreateObject	Creates a (local-server or remote-server) COM automation object instance
Exit	Exits script execution
LoadConfiguration	Loads a new configuration from the harddisk into uICE
Quit	Quits uICE
SetTimer	Set a timer
Sleep	Suspends script execution for a specified time
KillTimer	Cancels a timer

Properties

`ActiveApplication`	The name of the active Application Preset (read/write string)
`RemoteControl`	The remote control name of the currently executing command (read/write string). Initially, this property is set to the remote control of the executing command. After modifying it, all following operations (such as setting ActiveApplication) affect the settings of the new remote control.
`Status`	Status text in the status line of the uICE main window (read/write string)

Objects

Application	Properties and methods specific to executable files
Command	Properties of the currently executing command
Display	Display adapter operations
Globals	Access to global variables
Mixer	Audio control
Mouse	Mouse control
OSD	Programmable OSD
Power	Power functions like system shutdown and wake-up
Window	Properties and methods specific to windows

Alert method

Displays a text-only OSD

```
Alert( sText )
```

Arguments

sText
Required. Text string to display in the OSD. You can use common format characters like `\n` (line-break) and `\t` (tab).

Remarks

Same as calling `OSD.Show(osdText,sText)`.

See Also

OSD Object

CreateObject method

Creates and returns a reference to a COM automation object instance

```
CreateObject( sProgIDorClassID [, sLocation] )
```

Arguments

sProgIDorClassID
Required. Either a ProgID in the form *servername.typename*, or the CLSID of the object. See below for an example.

sLocation
Optional. The name of the network server where the object is to be created.

Remarks

This method creates an instance of the automation server specified in sProgIDorClassID. Only local-server or remote-server instances are created, i.e. automation servers running in its own executable.

sProgIDorClassID can be a ProgID, or a CLSID in the form "{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}".

JScript example:

```
wmp6=CreateObject("MediaPlayer.MediaPlayer"); //create an instance of Windows Media Player 6
wmp6.Filename="c:\\videos\\whatever.avi"; //load a media file.
wmp6.Play(); //Play it
```

sLocation can be the servername/sharename of a networked remote computer on which the object is to be created. Note that Windows NT networks have access restrictions, which can be configured using the *dcomcnfg.exe* tool. On the remote computer, go to Start Menu|Run, type *dcomcnfg* and click OK. Locate Component Services|Computers|My Computer, right click and select Properties, Default COM Security.

Exit method

Exits script execution.

```
Exit( [eExitCode] )
```

Arguments

eExitCode
Optional. An exit code value from the ExitCode enumeration. See below for possible values.

Remarks

eExitCode can be one of the following values:

`ExitCodeNoOSD`	Exit script and continue with the command's next action, but do not display a generic OSD for the command.
`ExitCodeNormal`	Exit script and continue with the command's next action. This is the default.
`ExitCodeAbort`	Exit script and abort further command processing, but display a generic command OSD.
`ExitCodeAbortNoOSD`	Exit script and abort further command processing without displaying any OSDs.

KillTimer method

Cancel a running timer

```
KillTimer( sTimerName )
```

Arguments

sTimerName
Required. The name of a running timer.

Remarks

This method cancels a timer that has been set previously by a call to SetTimer. The timer function will not be called, and the script that called SetTimer will be released.

This method can be called from any script, not only from the script that set the timer.

See Also

SetTimer

LoadConfiguration method

Displays a text-only OSD

```
LoadConfiguration( sConfigurationName )
```

Arguments

sConfigurationName
Required. Text string containing the name of an existing configuration.

Remarks

Configuration names are the same names as from the main menu's *Preferences*|*Configurations.*

See Also

Load configuration action

Quit method

Quits uICE

```
Quit()
```

SetTimer method

Set a timer

```
SetTimer( sTimerName, nTimeOut )
```

Arguments

sTimerName
Required. The timer name

nTimeOut
Required. TimeOut value in milliseconds.

Remarks

After a timer runs out (i.e. after nTimeOut milliseconds), a function in the script with the same name as sTimerName is called.

After calling the SetTimer method, the script remains in memory (including all variables) until the timer runs out or the KillTimer method is called.

JScript example:

```
TestVar="The Timer"; //set a string variable used later in the timer function

function MyTimer() //the timeout function. Note the name
{
    Alert(TestVar); //TestVar is still valid
}

SetTimer("MyTimer",2000); //set the timer to call MyTimer() after 2 seconds
```

Each script that uses SetTimer must have an appropriate timer function. The timer name is global and can be used in different scripts across commands and applications.

If you call SetTimer from two different scripts, the timer function in the last executed script will be called.
Let's say script A calls SetTimer. After script A has been executed, script B calls SetTimer with the same sTimerName as script A. This causes script A to be deleted from memory (including all its variables) and replaced by script B. The timer function of script B will be called after the timeout. If you need to pass variables from one script to another, use the Globals object.

See Also

KillTimer

Sleep method

Suspends script execution for a specified time

```
Sleep( nMilliSecs )
```

Arguments

nMilliSecs
Required. Amount of time, in milliseconds, to suspend execution.

Application Object

The Application object provides properties and methods specific to executable files

```
Application( [sNameOrFile] )
```

Arguments

sNameOrFile
Optional. Either the name of an existing Application Preset or a path and file name.

Remarks

If sNameOrFile is omitted, an object representing the application preset of the current command is returned. If the current command does not belong to an application preset, the returned application object is empty, i.e. its Name, File and Window properties contain only empty values.

If sNameOrFile is a path and/or filename, uICE searches the available application presets for a matching executable. If none is found, the returned object's Name property is an empty string, but all other properties and methods are valid on this object.

Click the icon from the scripteditor's toolbar to create an application object from a file.

See Also

Window Object

Properties

`Name`	Read-only name of the application. Only valid if the object represents an existing application preset.
`File`	The full path and filename of the application (read-only)
`Window`	The application's main window object (read-only). Only valid if the application is running and has at least one window.
`Running`	Read/write boolean value indicating whether the application is currently running.

Methods

`Execute( [sCmdLine] [, sVerb] )`

Executes the file contained in the File property. Same as setting the Running property to `true`, except you can optionally supply a command line and/or a shell verb

Arguments

sCmdLine
Optional command line

sVerb
Optional shell verb to perform on the file, if the file is not an executable (e.g. "open", "play", "print")

JScript example:

```
Application("notepad.exe").Execute("C:\\Autoexec.bat"); //open C:\Autoexec.bat in notepad
```

```
Application("Readme.txt").Running=true; //open Readme.txt in associated application (usually notepad)
```

`Close()`

Closes the application. Same as setting the Running property to `false`.

Command Object

Contains properties of the currently executing command, i.e the command executing the script.

`Command`

Properties

`Name`	The command's name
`Count`	Number of layouts (>= 1). Read-only.
`Layout( nIndex )`	Retrieves the name of the layout specified by `nIndex`. nIndex can be a number between zero and one less than Count.
`ActiveLayout`	Contains the index of the active layout. Read/Write, can be set to a valid layout index (0 <= index < Count) to switch the command to another layout.
`RepeatCount`	The repeat counter. RepeatCount is 0 for the first command and increases by 1 for each repeated command. Read-only.

Display Object

Display adapter methods and properties.

```
Display( [nDisplayNum] )
```

Arguments

nDisplayNum
Optional. The index number of the display adapter.
Valid numbers range from zero to one less of the amount of installed gfx cards.
The default is 0.

Properties

`Name`	Read-only string containing the name of the display adapter hardware.
`Number`	Read-only, zero-based index number containing the display adapter number.
`Off`	Read/write boolean value indicating the power management state of the display. Can be set to `true` to turn the display off.
`Width`	The actual screen width. Read-only.
`Height`	The actual screen height. Read-only.

Methods

```
SetMode( nWidth, nHeight, nBits, nFrequency )
```

Sets the display mode.

Arguments

nWidth
Required. Screen width

nHeight
Required. Screen height.

nBits
Required. Color depth in bits.

nFrequency
Required. Refresh rate in Hz.
Note that some display adapters do not support refresh rate manipulation.
Set nFrequency to 0 in this case.

JScript example:

```
Display.SetMode (1280,1024,32,85); // set the display mode to 1280x1024, 32 Bits, 85Hz
```

See Also

Change Display Mode action

Globals Object

The Globals object can be used to save variables between script executions or to pass variables to other scripts.

```
Globals( sVariableName, Value )
```

Arguments

sVariableName
Required. A string containing the name of the variable.

Value
Required.Variant value of the variable.

Remarks

The Globals collection object can store any kind of values. All variables remain in memory until uICE exits and are accessible from different scripts across commands and applications.

Use a somewhat unique name for sVariableName to prevent overwriting values of other scripts, unless this is intended of course.

To remove a variable from the object, call `Globals.Remove(sVariableName)`

VBScript example:

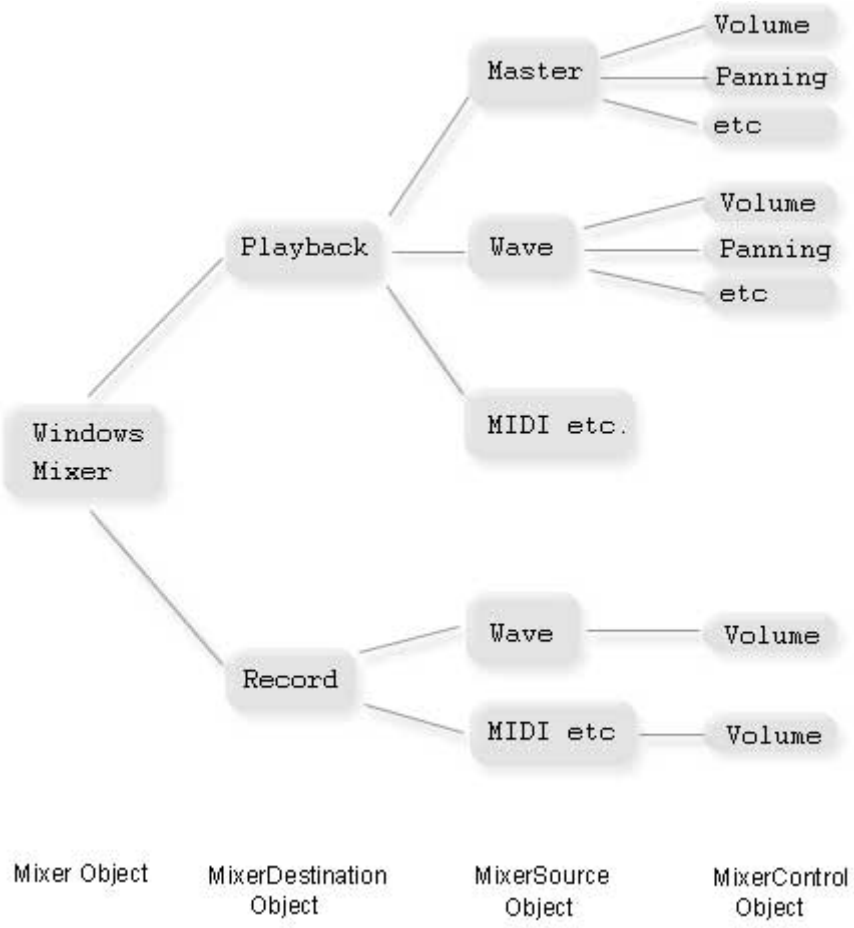
```
Globals("MyGlobalNumber")=2003
Globals("MyGlobalString")="uICE"

Alert Globals("MyGlobalString") + CStr(Globals("MyGlobalNumber"))

Globals.Remove "MyGlobalNumber"
```

Audio Mixer Overview

uICE provides a set of objects to represent the Windows audio mixer in an object oriented way. The following illustration shows a typical mixer hierarchy and its representing objects:



The Mixer object basically represents a single soundcard.

Common soundcards have two fixed "destinations", Playback and Record, represented by the MixerDestination object. Typically you won't use the MixerDestination object directly (except for enumeration purposes, if required), but index into a MixerSource object using `Mixer.Playback(sourceindex)` or `Mixer.Record(sourceindex)`.

A MixerSource is a single audio data source, like CD-IN, MIDI, Wave and so on. Each source has a set of MixerControls to adjust properties like Volume, Bass and Panning.

MixerDestinations and MixerSources are collection objects, i.e. you can enumerate the containing child objects.

Mixer Object

The Mixer object represents a single soundcard.

```
Mixer( [nMixerNum] )
```

Arguments

nMixerNum
Optional. Index number of the soundcard.
Values can range from 0 to one less than the amount of installed soundcards.
The default is 0.

Remarks

If you have more than one soundcard installed in your system, `Mixer(0)` is your first sound device, `Mixer(1)` the second and so on.

For the more common situation with a single soundcard, you can omit the nMixerNum parameter.

Methods

```
SetDefault()
```

Sets this mixer object as the default sound device in uICE. This affects all Audio Control actions.

See Also

Mixer Overview

MixerDestination Object

The MixerDestination object represents a collection of audio data sources. A destination is usually Playback or Record.

```
Mixer.Playback
```

```
Mixer.Record
```

```
Mixer.Destination( DestinationIndex )
```

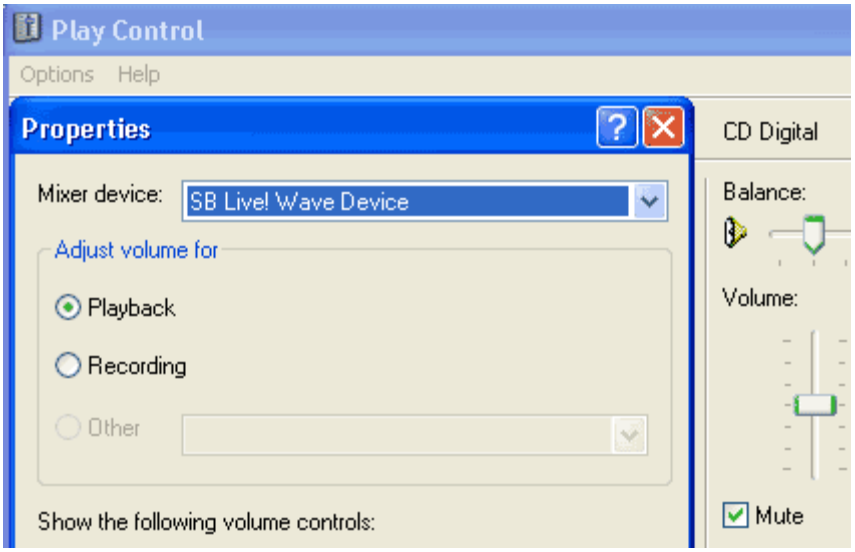
Arguments

DestinationIndex
Required. A destination index number (zero to Mixer.Count). DestinationIndex=0 is usually the same as Mixer.Playback, DestinationIndex=1 is Mixer.Record.

Remarks

The Destination property can be used to access additional, non-standard mixer destinations besides the default Playback and Record.

The following image shows the equivalent settings in the properties of the Windows Mixer control panel applet. The destinations can be seen under *Adjust volume for*, where the *Playback* radio button is equal to the Mixer.Playback destination, *Recording* to Mixer.Record and *Other* to Mixer.Destination(Index).



See Also

MixerSource Object

MixerSource Object

The MixerSource object represents a single audio data source.

```
Mixer.Playback( SourceIndexOrName )
```

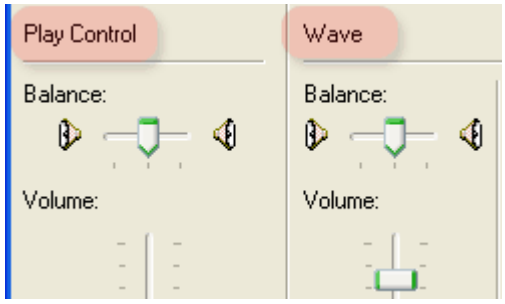
```
Mixer.Record( SourceIndexOrName )
```

Arguments

SourceIndexOrName
Required. Either an index number or source name. See remarks.

Remarks

The SourceIndexOrName parameter can be either a numerical index or a name of a valid mixer source. Mixer source names are the labels that appear in the Windows volume control panel applet. The following image shows an example of the two first mixer sources of a playback destination.



Play Control can be accessed with `Mixer.Playback(0)` or `Mixer.Playback("Play Control")`.

The Wave source is `Mixer.Playback(1)` or `Mixer.Playback("Wave")`.

MixerSource objects are collection objects containing one or more MixerControl objects. The following VBScript example enumerates all available MixerControl objects of the "Wave" MixerSource:

```
Dim str
For Each control In Mixer.Playback("Wave")
  str=str+control.Name+" "
Next

Alert str 'str contains "Volume Mute "
```

or using a JScript `for(;;)` loop and a numerical index:

```
str="";
for (i=0; i<Mixer.Playback(1).Count; i++)
  str=str+Mixer.Playback(1)(i).Name+" ";

Alert(str);
```

See Also

MixerControl Object

MixerControl Object

The MixerControl object contains properties and methods of a single mixer control like volume or mute.

```
Mixer.mixerdestination( SourceIndex ).Control( ControlIndexOrName )
```

```
Mixer.mixerdestination( SourceIndex )( ControlIndexOrName )
```

Arguments

ControlIndexOrName
Required. Numerical index or name of the control.

```
Mixer.mixerdestination( SourceIndex ).Volume
```

```
Mixer.mixerdestination( SourceIndex ).Treble
```

```
Mixer.mixerdestination( SourceIndex ).Bass
```

```
Mixer.mixerdestination( SourceIndex ).Panning
```

```
Mixer.mixerdestination( SourceIndex ).Fader
```

```
Mixer.mixerdestination( SourceIndex ).Mute
```

Arguments

mixerdestination
Required. A MixerDestination object (i.e. `Playback` or `Record`)

SourceIndex
Required. Index to a MixerSource object.

Properties

`Name`	Read-only string containing the name of the object
`Value`	Read/write long value containing the value of the object. The value is a combination of all channels. Valid values depend on the type of the control.
`Channels`	Number of channels. 2=Stereo, 1=Mono
`Channel`	Read/write number containing the value of a single channel.

Remarks

The Value property is a combination of all channels. Value is the default property of the MixerControl object. Examples of valid Values:

MixerControl type volume, bass or treble: 0 to 100 in percent

MixerControl type panning or fader: -100 to 100 in percent (negative is left or rear, 0 is center)

MixerControl type mute (boolean): 0 or 1 (false/true)

Examples:

```
Mixer.Playback(0).Volume = 10 // set volume to 10%
Mixer.Playback(0).Panning = -50 // set panning to 50% left
Mixer.Playback(0).Mute = true // mute
Mixer.Playback(0).Mute = !Mixer.Playback(0).Mute.Value//toggle mute
```

Methods

```
mixercontrol.ShowOSD()
```

Displays an appropriate OSD for the object.

Arguments

mixercontrol
Required. A MixerControl object (see above)

JScript example:

```
Mixer.Playback(0).Volume.ShowOSD();
```

Mouse Object

The Mouse object provides properties and methods related to the mouse cursor and mouse buttons.

```
Mouse
```

Properties

X	Read/write unsigned horizontal coordinate of the mouse cursor
Y	Read/write unsigned vertical coordinate of the mouse cursor
Button(`[eMouseButton=btnLeft] [, nXButton=0]` **)**	Read/write state of the mouse button specified in eMouseButton and nXButton. See remarks.

Remarks

eMouseButton can have one of the following values:

`btnLeft`	Left mouse button. The default.
`btnMiddle`	Middle mouse button
`btnRight`	Right mouse button
`btnX`	Extended (X) mouse button specified in nXButton. `nXButton=0` or `nXButton=1`: first X button (default), `nXButton=2`: second X button

The Button property can have either the value `dirDown` (1) or `dirUp` (0).

X and Y are screen coordinates in pixel, i.e. in 1280x1024 display mode, the following example clicks the right mouse button in the lower right corner of the screen (note coordinates are zero-based):

```
Mouse.X=1279
Mouse.Y=1023
Mouse.Button(btnRight)=dirDown
Mouse.Button(btnRight)=dirUp //don't forget to release the button!
```

Use the Display object to determine the current screen resolution.

Click the icon in the script editor's toolbar to insert mouse coordinates automatically.

Methods

```
Wheel( eDirection [, delta=120] )
```

Scrolls the mouse wheel.

Arguments

eDirection
Required. `dirUp` or `dirDown`

delta
Optional.
Unsigned number representing the amount of wheel movement.
The default is 120, which is equal to one wheel click.

```
ButtonDown( eMouseButton=btnLeft [, nXButton=0] )
```

```
ButtonUp( eMouseButton=btnLeft [, nXButton=0] )
```

Same functionality as the Button property. See above for a description of arguments.

OSD Object

The OSD object provides access to programmable On-Screen Displays

```
OSD
```

Properties

`Visible`	Read-only boolean value indicating whether the OSD is currently visible
`Type`	Read/write value from the eOSDType enumeration indicating the current OSD type. See remarks.
`TimeOut`	Read/write unsigned long value containing the OSD timeout, in milliseconds. Don't forget to restore it to its original value after modifying it.

Methods

```
Show( eOSDType, Param1 [, Param2] )
```

Displays the OSD.

Arguments

eOSDType
Required. Value from the eOSDType enumeration. See remarks.

Param1
Required. Variant value depending on eOSDType. See remarks.

Param2
Required/Optional. Variant value depending on eOSDType. See remarks.

Remarks

The Type property and the eOSDType parameter sets the type of the OSD that is to be displayed. It can be one of the following constants:

`osdVolume`	Displays the volume OSD. `Param1` is a string that is to be displayed above the graphical representation of the current volume. `Param2` is an unsigned number containing the current volume in percent (0..100)
	Displays the Treble OSD.

`osdTreble`	`Param1` is a string that is to be displayed above the graphical representation of the current treble value. `Param2` is an unsigned number containing the current treble value in percent (0..100)
`osdBass`	Displays the Bass OSD. `Param1` is a string that is to be displayed above the graphical representation of the current bass value. `Param2` is an unsigned number containing the current bass value in percent (0..100)
`osdPanning`	Displays the Panning OSD. `Param1` is a string that is to be displayed above the graphical representation of the current Panning value. `Param2` is an signed number containing the current Panning value in percent (-100..+100).
`osdFader`	Displays the Fader OSD. `Param1` is a string that is to be displayed above the graphical representation of the current Fader value. `Param2` is a signed number containing the current Fader value in percent (-100..+100)
`osdMute`	Displays the Volume OSD with a Mute icon in the upper right corner. `Param1` is a string that is to be displayed above the graphical representation of the current volume. `Param2` is an unsigned number containing the current volume in percent (0..100)
`osdProgress`	Displays a progress meter similar to the Volume OSD, but without icon. `Param1` is a string that is to be displayed above the progress meter. `Param2` is an unsigned number containing the progress value in percent (0..100)
`osdText`	Displays a text OSD. Similar to calling the Alert method. `Param1` can be a string or a Window object. In the latter case, the window's title and its icon is shown. `Param2` is ignored.
`osdHTML`	Displays HTML formatted text. `Param1` is a string containing the HTML portion between the <BODY></BODY> tags (without these tags). `Param2` is ignored.

JScript examples:

```
OSD.Show(osdVolume,Mixer.Playback(0).Name+" Volume",Mixer.Playback(0).Volume);
    //Same as Mixer.Playback(0).Volume.ShowOSD();

OSD.Show(osdMute,Mixer.Playback(0).Name+" Volume",Mixer.Playback(0).Volume);
   //Same as Mixer.Playback(0).Mute.ShowOSD();

OSD.Show(osdText,"Hello World");
   //Same as Alert("Hello World");

OSD.Show(osdHTML,"<p align='center'><b><i>Hello World</i></b></p>");
```

Power Object

The Power object contains power management related properties and methods

Power

Properties

`TimeOut`	Read/write timeout value, in seconds, for all power management methods.
`WakeTime`	Read/write string containing the system wake-up time. The string must be in the format hh:mm either in 24 hour or 12 hour format with PM/AM. To cancel a previously set wake-up time, set WakeTime to an empty string, i.e. `Power.WakeTime=""`. Example: `Power.WakeTime="23:59"` resumes the system at 1 minute before midnight. `Power.WakeTime="10:00PM"` wakes at 10pm.

Methods

```
Shutdown( nTimeOut )
```

```
Suspend( nTimeOut )
```

```
Hibernate( nTimeOut )
```

```
Reboot( nTimeOut )
```

System shutdown, standby mode, hibernate mode and reboot.

Arguments

nTimeOut
Required. Timeout value in seconds after which the respective function is invoked.

Remarks

If nTimeOut is less than or equal to 15 seconds (or reaches 15 seconds), an OSD countdown is displayed. Use 0 for nTimeOut to perform the respective function immediately.

To cancel pending power methods, set the Timeout property to 0, i.e. `Power.TimeOut=0`

See Also

Shutdown Windows, Reboot Windows, Suspend Windows actions

On-Screen Menu Timers

Window Object

The Window object represents an existing desktop window.

```
Window( [sWindowPath] )
```

Arguments

sWindowPath
Optional. Text string containing the Window Path to the window.
If omitted, a window object representing the current foreground window is returned.

Remarks

Click the icon in the scripteditor's toolbar to create a Window object from an existing window.

Properties

`hWnd`	Win32 window handle of the window
`Path`	Read-only string containing the window path
`Name`	Read-only string containing the window's title
`ClassName`	Read-only string containing the window's class name
`X`	Read/write unsigned value representing the horizontal position of the upper left corner of the window
`Y`	Read/write unsigned value representing the vertical position of the upper left corner of the window
`Width`	Read/write unsigned width of the window
`Height`	Read/write unsigned height of the window
`Minimized`	Read/write boolean value indicating whether the window is minimized
`Maximized`	Read/write boolean value indicating whether the window is maximized
`Visible`	Read/write boolean value indicating whether the window is visible
`Application`	Read-only string containing the filename of the executable that owns the window.
`Parent`	The window's parent window object (read-only)

Remarks

You can resize, move, minimize, maximize and/or hide windows by setting the above properties.

Methods

Close	Closes the window.
SwitchTo	Puts the window into foreground.
SetMousePosition	Sets the mouse cursor relative to the window's upper left corner.
KeyStroke	Type into the window.
KeyDown KeyUp	Presses a key and holds it down (KeyDown) or releases a key (KeyUp).
PostMessage SendMessage	Posts or sends a message to the window.
Window	Returns a window object related to this window.
Refresh	Refreshes the window object and updates internal window caches.

Close()

Closes the window.

SwitchTo()

Puts the window into foreground.

SetMousePosition(*x,y*)

Sets the mouse cursor relative to the window's upper left corner.

Arguments

x,y
Required. x and y coordinate relative to the window's upper left corner

Remarks

Click the icon in the scripteditor's toolbar to insert a SetMousePosition method with proper coordinates automatically.

Keystroke(*sKeys*)

Type into the window

Arguments

sKeys
Required. String containing keystrokes. See Keystroke actions.

Remarks

Click the icon in the scripteditor's toolbar to insert keystrokes.

KeyDown(*eVkCode*)

KeyUp(*eVkCode*)

Presses a key and holds it down (KeyDown) or releases a key (KeyUp).

Arguments

eVkCode
Required. Virtual KeyCode value from the eVkCodes enumeration.

Remarks

Never forget to call KeyUp after KeyDown.

```
PostMessage( nMsg, wParam, lParam )
```

```
SendMessage( nMsg, wParam, lParam )
```

Posts or sends a message to the window.

Arguments

nMsg
Required. WM_xxx value from the eWinMessages enumeration, or a numerical value representing a valid Win32 message.

wParam
Required. Appropriate WPARAM value for nMsg. Depends on the message.

lParam
Required. Appropriate LPARAM value for nMsg. Depends on the message.

Remarks

PostMessage and SendMessage behave exactly like their Win32 API equivalents. SendMessage can return values from the window's window procedure to the script.

Click the icon in the scripteditor's toolbar to insert a PostMessage method from a captured message.

```
Window( eWindow )
```

Returns a window object related to this window, i.e. a child window, the next window in the Z order, the top level parent and so on.

Arguments

nMsg
Required. wndXXX value from the eWindow enumeration. See remarks.

Remarks

eWindow can be one of the following constants:

`wndFirst`	The sibling window that is on top of the Z order
`wndLast`	The sibling window that is below all others in the Z order
`wndNext`	Next sibling window in the Z order
`wndPrevious`	Previous sibling window in the Z order
`wndOwner`	The window's owner window, if any
`wndChild`	The first child window of this window, if any
`wndTop`	Same as wndFirst
`wndTopLevel`	The top level parent window

Refresh(*sNewWindowPath* **)**

Refreshes the window object and updates internal window caches using the the window path specified in sNewWindowPath.

sNewWindowPath
Required. The new window path for the object.

See Also

Introduction to window messages

Window Paths

Keystroke actions

eVkCodes enumeration

eWinMessages enumeration

eVkCodes enumeration

The following virtual key code constants are available to scripts:

```
typedef enum tagVkCodes
{

     VK_LBUTTON =0x01,
     VK_RBUTTON =0x02,
     VK_CANCEL =0x03,
     VK_MBUTTON =0x04,
     VK_XBUTTON1 =0x05,
     VK_XBUTTON2 =0x06,
     VK_BACK =0x08,
     VK_TAB =0x09,
     VK_CLEAR =0x0C,
     VK_RETURN =0x0D,
     VK_SHIFT =0x10,
     VK_CONTROL =0x11,
     VK_MENU =0x12,
     VK_ALT =0x12,
     VK_PAUSE =0x13,
     VK_CAPITAL =0x14,
     VK_KANA =0x15,
     VK_HANGEUL =0x15,
     VK_HANGUL =0x15,
     VK_JUNJA =0x17,
     VK_FINAL =0x18,
     VK_HANJA =0x19,
     VK_KANJI =0x19,
     VK_ESCAPE =0x1B,
     VK_CONVERT =0x1C,
     VK_NONCONVERT =0x1D,
```

```
VK_ACCEPT =0x1E,
VK_MODECHANGE =0x1F,
VK_SPACE =0x20,
VK_PRIOR =0x21,
VK_NEXT =0x22,
VK_END =0x23,
VK_HOME =0x24,
VK_LEFT =0x25,
VK_UP =0x26,
VK_RIGHT =0x27,
VK_DOWN =0x28,
VK_SELECT =0x29,
VK_PRINT =0x2A,
VK_EXECUTE =0x2B,
VK_SNAPSHOT =0x2C,
VK_INSERT =0x2D,
VK_DELETE =0x2E,
VK_HELP =0x2F,
VK_LWIN =0x5B,
VK_RWIN =0x5C,
VK_APPS =0x5D,
VK_SLEEP =0x5F,
VK_NUMPAD0 =0x60,
VK_NUMPAD1 =0x61,
VK_NUMPAD2 =0x62,
VK_NUMPAD3 =0x63,
VK_NUMPAD4 =0x64,
VK_NUMPAD5 =0x65,
VK_NUMPAD6 =0x66,
VK_NUMPAD7 =0x67,
VK_NUMPAD8 =0x68,
VK_NUMPAD9 =0x69,
VK_MULTIPLY =0x6A,
VK_ADD =0x6B,
VK_SEPARATOR =0x6C,
VK_SUBTRACT =0x6D,
VK_DECIMAL =0x6E,
VK_DIVIDE =0x6F,
VK_F1 =0x70,
VK_F2 =0x71,
VK_F3 =0x72,
VK_F4 =0x73,
VK_F5 =0x74,
VK_F6 =0x75,
VK_F7 =0x76,
VK_F8 =0x77,
VK_F9 =0x78,
VK_F10 =0x79,
VK_F11 =0x7A,
VK_F12 =0x7B,
VK_F13 =0x7C,
VK_F14 =0x7D,
VK_F15 =0x7E,
VK_F16 =0x7F,
VK_F17 =0x80,
VK_F18 =0x81,
VK_F19 =0x82,
VK_F20 =0x83,
VK_F21 =0x84,
VK_F22 =0x85,
```

```
VK_F23 =0x86,
VK_F24 =0x87,
VK_NUMLOCK =0x90,
VK_SCROLL =0x91,
VK_OEM_NEC_EQUAL =0x92,
VK_OEM_FJ_JISHO =0x92,
VK_OEM_FJ_MASSHOU =0x93,
VK_OEM_FJ_TOUROKU =0x94,
VK_OEM_FJ_LOYA =0x95,
VK_OEM_FJ_ROYA =0x96,
VK_BROWSER_BACK =0xA6,
VK_BROWSER_FORWARD =0xA7,
VK_BROWSER_REFRESH =0xA8,
VK_BROWSER_STOP =0xA9,
VK_BROWSER_SEARCH =0xAA,
VK_BROWSER_FAVORITES =0xAB,
VK_BROWSER_HOME =0xAC,
VK_VOLUME_MUTE =0xAD,
VK_VOLUME_DOWN =0xAE,
VK_VOLUME_UP =0xAF,
VK_MEDIA_NEXT_TRACK =0xB0,
VK_MEDIA_PREV_TRACK =0xB1,
VK_MEDIA_STOP =0xB2,
VK_MEDIA_PLAY_PAUSE =0xB3,
VK_LAUNCH_MAIL =0xB4,
VK_LAUNCH_MEDIA_SELECT =0xB5,
VK_LAUNCH_APP1 =0xB6,
VK_LAUNCH_APP2 =0xB7,
VK_OEM_1 =0xBA,
VK_OEM_PLUS =0xBB,
VK_OEM_COMMA =0xBC,
VK_OEM_MINUS =0xBD,
VK_OEM_PERIOD =0xBE,
VK_OEM_2 =0xBF,
VK_OEM_3 =0xC0,
VK_OEM_4 =0xDB,
VK_OEM_5 =0xDC,
VK_OEM_6 =0xDD,
VK_OEM_7 =0xDE,
VK_OEM_8 =0xDF,
VK_OEM_AX =0xE1,
VK_OEM_102 =0xE2,
VK_ICO_HELP =0xE3,
VK_ICO_00 =0xE4,
VK_PROCESSKEY =0xE5,
VK_ICO_CLEAR =0xE6,
VK_OEM_RESET =0xE9,
VK_OEM_JUMP =0xEA,
VK_OEM_PA1 =0xEB,
VK_OEM_PA2 =0xEC,
VK_OEM_PA3 =0xED,
VK_OEM_WSCTRL =0xEE,
VK_OEM_CUSEL =0xEF,
VK_OEM_ATTN =0xF0,
VK_OEM_FINISH =0xF1,
VK_OEM_COPY =0xF2,
VK_OEM_AUTO =0xF3,
VK_OEM_ENLW =0xF4,
VK_OEM_BACKTAB =0xF5,
VK_ATTN =0xF6,
```

```
     VK_CRSEL =0xF7,
     VK_EXSEL =0xF8,
     VK_EREOF =0xF9,
     VK_PLAY =0xFA,
     VK_ZOOM =0xFB,
     VK_NONAME =0xFC,
     VK_PA1 =0xFD,
     VK_OEM_CLEAR =0xFE

} eVkCodes;
```

eWinMessages enumeration

The following Win32 message constants are available to scripts:

```
typedef enum tagWinMessages
{

     WM_INPUT = 0x0FF,
     WM_KEYDOWN = 0x100,
     WM_KEYUP = 0x101,
     WM_CHAR = 0x0102,
     WM_COMMAND = 0x0111,
     WM_SYSCOMMAND = 0x112,
     WM_LBUTTONDOWN = 0x0201,
     WM_LBUTTONUP = 0x202,
     WM_LBUTTONDBLCLK = 0x0203,
     WM_RBUTTONDOWN = 0x0204,
     WM_RBUTTONUP = 0x0205,
     WM_RBUTTONDBLCLK = 0x0206,
     WM_MBUTTONDOWN = 0x0207,
     WM_MBUTTONUP = 0x0208,
     WM_MBUTTONDBLCLK = 0x0209,
     WM_MOUSEWHEEL = 0x020A,
     WM_XBUTTONDOWN = 0x020B,
     WM_XBUTTONUP = 0x020C,
     WM_XBUTTONDBLCLK = 0x020D,
     WM_HOTKEY = 0x0312,
     WM_APPCOMMAND = 0x0319,
     WM_USER = 0x0400,
     WM_APP = 0x8000

} eWinMessages;
```